Design of a M2M waste container system for a Smart City

by

Cristian Moreno Ruiz

A Master's Thesis Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

In partial fulfilment of
the requirements for the

Master in Telecommunications Engineering

Advisor:
Anna Calveras Auge

Barcelona, July 2016

Abstract

One of the major challenges of a city or town is the management of waste collection routes, so that containers are not overfilled or collected unnecessarily. So as to make this task easier, a system capable of reporting the fill levels of a waste container is presented in this thesis, so collection routes can be planned efficiently with live information. Moreover, it incorporates additional sensors to detect vandalism acts, such as container displacements or fire.

The work in this thesis involves research about distance measurement algorithms, and hardware components needed for the system measurements, as well as a hands on prototype implementation, featuring fill level, acceleration and flame presence measurements. Then, the performance of the prototype is analysed, discussing the suitability of the hardware components used.

Acknowledgements

In the first place, I would like to thank my advisor Anna Calveras, for trusting me to carry out this thesis, and also for her advice, knowledge and continuous support during my master's thesis.

Secondly, I would like to thank Carlos and Quim, for helping me with the circuit designs and measurements for the power analysis.

Finally, I would like to thank also my family, friends and mates who have encouraged me during all my years of study until today.

Revision history and approval record

Revision	**Date**	**Purpose**
0	29/01/2016	Document Creation. Chapter 1 added
1	23/06/2016	Chapters 2 and 3 added
2	06/07/2016	Chapters 4, 5, 6 and 7 added
3	12/07/2016	Final Document reviewed

Name	**e-Mail**
Cristian Moreno Ruiz	cristianmr.28@gmail.com
Anna Calveras Auge	anna.calveras@entel.upc.edu

Written by:		**Reviewed and approved by:**	
Date	12/07/2016	Date	13/07/2016
Name	Cristian Moreno Ruiz	Name	Anna Calveras Auge
Position	Project Author	Position	Project Supervisor

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This Master's thesis is carried out at the department of Telematics of the "Universitat Politècnica de Catalunya". The project aims at designing and implementing a hands-on prototype of a system to improve the labour of collecting the waste of a Smart City, by means of a Machine to Machine (M2M) system that monitors the fill levels of the waste containers. Moreover, this system must be also capable of detecting vandalism acts such as fire, so that alerts can be programmed in case vandalism is detected.

In this chapter, a description of the purpose and the scope of the project are provided, as well as the detail of the work plan followed. The project is divided in four work packages described here, and a Gantt diagram illustrates the time plan.

1.1 Purpose of the Thesis

Waste collection management in cities is a challenge that can become problematic if it is not handled efficiently. It involves a great number of containers distributed all over the city, which are usually filled at a different and unpredictable velocity, and they have to be collected periodically. With such number of distributed containers and the variety of their filling rate, their collection routes management becomes a very challenging task. The resulting collection routes are usually not accurate enough to provide a good service, causing both overfilled containers not being collected, and unnecessary collections of empty containers.

Moreover, most of the vandalism acts in the cities are performed over the waste containers, either moving or burning them. It involves an additional cost for the waste management, which needs to be notified about moved or burnt containers, and collect and replace them.

In this thesis, a waste monitoring system is proposed capable of reporting the fill level of the waste containers over the General Packet Radio Service (GPRS) network, as well as

other information about its status such as any acceleration registered or the presence of fire. With the use of this system, it is possible to determine if a container needs to be emptied or not, thus improving the efficiency of the collection routes by saving fuel, CO_2 emissions and traffic. Moreover, it allows a premature detection of vandalism acts performed over a bin.

As the ultimate feature, the system is intended to include Bluetooth Low Energy (BLE) connectivity, so that data from nearby sensors can be collected and reported over the GPRS network. This is the main feature that differentiates this system from the existing ones, which allows decreasing the number of needed Subscriber Identity Module (SIM) contracts for nearby containers, or being integrated with other sensors without GPRS connectivity, thus making the difference towards a fully connected Smart City.

Figure 1.1 illustrates the architecture and features provided by the proposed system.

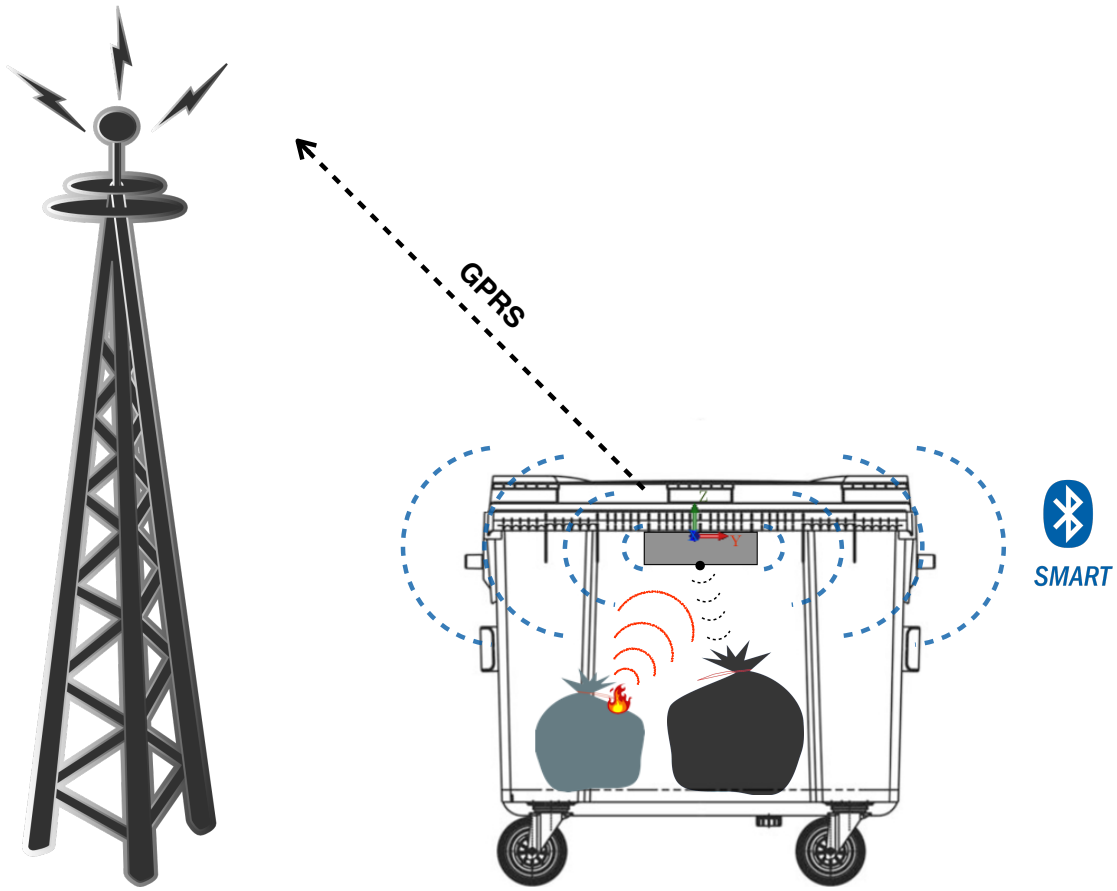


Figure 1.1: Overall architecture of the system

1.1.1 Scope of the project

The purpose of this thesis is to study the requirements of the system and start the design and the implementation of it. It means that a hands-on prototype is implemented, which includes the integration of the Microcontroller Unit (MCU) with the external ultrasonic, accelerometer and flame sensors.

However, it is a huge system which is not completely finished in this thesis, but may lead to be finished later. For this reason, huge documentation about the research and development process followed has to be done, in order to make the future work easier. Moreover, a complete description of what is already implemented and which should be the next steps is needed.

1.2 System requirements

The required system must be capable of detecting the fill level of a container, by means of ultrasonic sensors. Different algorithms to estimate the real level using ultrasound techniques are described in chapter 2, identifying the most suitable method, in terms of cost and installation feasibility for the purpose of the project.

Moreover, the system must be able to detect vandalism acts such as fire or any movement of the containers, for which purpose, other sensors like a near infra-red flame detectors and an accelerometer are also needed.

About the communications of the system, the collected data is intended to be transmitted to a remote server through the GPRS network, so integration with a GPRS module is required. In addition to GPRS connection, the system must include a BLE transceiver, so as to provide connectivity with nearby sensors, and allow the collection and reporting of the data collected by them.

As the brain of the system, a MCU must be used and programmed to perform the integration between all the sensors and peripherals. The MCU is in charge of acquiring data from all the integrated sensors, processing the data and transmitting it, if necessary, through the communication interfaces.

Finally, this system has to remain installed in a fixed location for a long time. For this reason, an important requirement of the whole system is to have a low power consumption, so as to provide a long battery lifetime, and a low maintenance cost.

1.3 Work plan

The work in this thesis has been divided in four Work Packages (WP) involving different tasks. These WPs are described below:

- Project Management: the tasks regarding to the writing of the different documents and the decisions taken during the project.
- Study of the State of the Art: the research about similar existing products already in the market, and the hardware and algorithms to be used by the system.
- System Implementation: the implementation of the system to determine the container fill-level and other measurements.
- System Performance Analysis: the analysis of the performance obtained with each implemented module.

These WPs are divided in several tasks and milestones, and they are detailed below.

<table>
<tr><td>Project Management</td><td colspan="2">WP1</td></tr>
<tr><td>Major constituent: documentation</td><td colspan="2" rowspan="2">Planned start date: 01/01/2016
Planned end date: 10/07/2016</td></tr>
<tr><td>Project planning and documentation deliverables.</td></tr>
<tr><td>Internal task T1.1: Project Planning.
Internal task T1.2: Review of the project goals achieved and reconsideration of them.
Internal task T1.3: Final Document writing.
Internal task T1.4: Defense of the thesis.</td><td>Deliverables:
Final Document</td><td>Dates:
10/07/2016</td></tr>
</table>

Table 1.1: WP1: Project Management

Project Researching	**WP2**	
Major constituent: researching	Planned start date: 18/01/2016 Planned end date: 15/04/2016	
Research of information about the topic: Similar existing products, ultrasonic distance determination algorithms and hardware for the system implementation.		
Internal task T2.1: To research about similar existing products. **Internal task T2.2:** To research about algorithms for fill level measurement by means of ultrasonic sensors. **Internal task T2.3:** To analyse available hardware components and its suitability for the system. **Internal task T2.4:** To read and understand documentation about the used hardware for implementation purposes.	**Deliverables:** None	**Dates:** None

Table 1.2: WP2. Project Researching

Prototype Implementation	**WP3**	
Major constituent: software development	Planned start date: 14/03/2016 Planned end date: 17/06/2016	
Implementation of the hands-on prototype integrating a MCU with external sensors and modules, involving mostly software development and some hardware development.		
Internal task T3.1: To program the MCU to read data from external sensors. **Internal task T3.2:** To design and build the electric circuit needed to connect the sensors with the MCU. **Internal task T3.3:** To implement new improvements arisen after first tests.	**Deliverables:** None	**Dates:** None

Table 1.3: WP3. Prototype Implementation

Prototype Performance Evaluation	**WP4**	
Major constituent: testing of the prototype.	Planned start date: 16/05/2016	
Tests execution in controlled scenarios, in order to evaluate the performance achieved in the prototype.	Planned end date: 17/06/2016	
Internal task T4.1: To execute tests for each sensor, and evaluate the accuracy and reliability of the measured data. **Internal task T4.2:** To propose possible improvements of the system, and evaluate the ones implemented.	**Deliverables:** None	**Dates:** None

Table 1.4: WP4. Prototype Performance Evaluation

WP#	**Task#**	**Title**	**Date**
WP3	T1	Firmware of MCU integration with sensors	13/05/2016
WP4	T2	Performance analysis and future improvements definition	06/06/2016
WP1	T3	Final Document	10/07/2016

Table 1.5: Milestones

A time plan has been designed in order to organize all the internal tasks involved in the WPs described before. The following Gantt Diagram illustrates the time plan followed in this thesis.

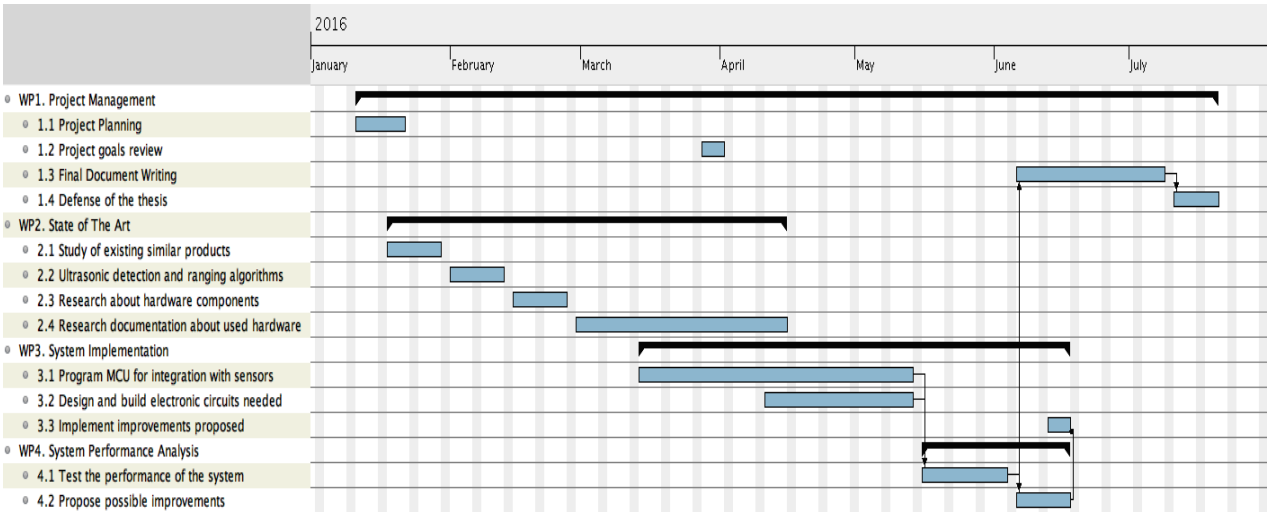


Figure 1.2: Gantt Diagram

Most of the time plan has been followed without deviations from the original one, but a problem was found with one of the ultrasonic sensors when performing the tests. The only SRF08 available was malfunctioning, so not all the tests have been performed successfully with this sensor.

Chapter 2

State of the Art

Ultrasonic sensors are the most used for fill level determination of waste containers. However, other methods like optical sensors can be used for this purpose. In this chapter, some systems oriented to the management of waste collection routes are compared.

Moreover, the basics on ultrasonic distance determination are presented followed by an analysis of some algorithms that may be used in this thesis for the fill level measurement. Finally, the hardware components needed for the prototype implementation are described, and some candidate components are presented.

2.1 Existing products

Recently, several systems have arisen that aim at helping the labour of managing the waste of a city. Most of those systems are capable of determining the fill level of a container and report it to a remote server, in order to use this information for collection routes management. Even though all of them have the same goal, each implementation varies and incorporates different sensors to provide additional data.

This section summarizes the peculiarities of those products that are more similar to the one presented in this thesis, and compares their features and modes of operation.

2.1.1 sigrenEa (aEnergis)

One of the pioneer products in the ambit of the monitoring of the waste container fill level is sigrenEa [21] [22]. They began in 2009, when this company (at first called aEnergis) launched an innovative system that was capable of detecting the fill level of trash containers,

and send the data periodically to a remote server via the GPRS network. In this way, the data can be used to plan the collection routes and manage the resources efficiently.

Their product was specifically designed to be robust, and it is fully recyclable and waterproof. It uses an ultrasonic sensor to determine the fill level of waste containers, and it is available in four different packaging options, depending on the type of container where it is going to be installed (below-ground, underground, textile or small indoor containers).

Regarding the communications, it is equipped with a Dual-Band GPRS module for long range communications, and also with a WM-BUS (Wireless Meter-BUS) module, which makes possible short range communication with specific hardware provided by the company. Optionally, the system can be complemented with a Global Positioning System (GPS) receiver, in order to maintain a precise location of the container.

The system does not need specific network configuration in the deployment phase, and it disposes of small batteries that have a different lifetime expectation depending on the chosen package, which can last from 5 to 12 years without replacement. In any case, the built-in batteries can be replaced easily if it is necessary.

2.1.2 Mobile Automation AG

Mobile Automation AG (MOBA) is an engineering company since 1972 and, as its name suggests, they offer several automated solutions in different sectors of the market. One important application is the waste management, and they have a huge portfolio of services destined for this sector.

One of the products in their portfolio is the filling level sensor (FLS-100) [10] [15], used to monitor and control the fill level of a waste container. By means of an ultrasonic sensor, the system is able to determine the fullness of the container, and it is reported over the GPRS network to plan an efficient recollection.

For this purpose, they use a Pepperl+Fuchs' ultrasonic sensor, which claims to be reliable and intelligent enough to adapt to changing surfaces and different kind of waste. The sensor is also equipped with a SIM card and it is able to send data to the cloud through the GPRS network, which also allows maintenance and diagnostics tasks.

The sensor has been designed to consume very low power, in order to make it a durable solution. It uses a lithium battery, which lasts up to ten years in operation. The solution, which has already been deployed in Barcelona, uses a software platform to provide an overview of the fill level of all the containers in a traffic light system, and also the specific filling percentage of each one.

2.1.3 Compology

In 2012, a start-up was built with the aim of developing a smart dynamic routing system for the waste industry; its name is Compology [1] [11]. They created WasteOS, which is composed of a wireless sensor, a routing software built and designed for the drivers, and an analysis tool made for tracking the performance metrics.

Their sensor is rugged in order to withstand the work conditions, and it uses a different technology from their competitors: a camera. The system takes a snapshot of the content of the bin, and it sends the image through the cell network, to the cloud. Then, a set of content parameters are extracted and analysed in order to obtain the fill level of the waste container.

Figure 2.1: Compology system including a camera. Source: [1]

Compology is a company that focuses in the interpretation of the obtained data, providing a dashboard that allows customers to easily monitor volume trends, without the need of being an expert in the matter. They also provide hardware to be used by the drivers, in order to obtain the most efficient collection routes they have to follow.

In 2014, the founders of Compology presented a patent application [3]. In this patent, they claim an invention of a method for waste management that includes recording an image of the content within a waste container, extracting a set of parameters to characterize the content of the trash container, and routing a waste removal vehicle based on the characterization of the content.

The most interesting and innovative feature reflected in this invention, is the capability of determining the actual contents of a waste container and its potential value, previously to the collection process. In addition to the quantity of waste allocated in a container, using the camera of the device it is possible to determine which are the contents of the bin. Combining this information with auxiliary sensors that detect other properties such as pH, density or temperature, they are able to generate specific information about the value of the waste before collecting it, thus allowing a really efficient planning of the collection routes.

2.1.4 SmartBin

SmartBin was founded in 2010, and it is one of the companies leading the sector of intelligent remote monitoring systems for the waste and recycling sectors. They offer products to manage different types of waste, from general waste and recyclables to fresh or used cooking oil.

Using an ultrasonic level sensor packed in a device that they call UBi [23], they collect reliable data and send it to the SmartBin Live platform. This platform offers to the user a clear view of the operations, and it creates optimized collection routes.

The UBi device has an internal GPRS antenna and a SIM card to report collected data via the cellular network. Moreover, it embeds a GPS module to locate the container, and also temperature and tilt sensors. The sensor is plug and play, it does not need maintenance and it has an embedded battery with ten years lifetime.

2.1.5 Enevo

Founded in 2010, Enevo is able to create efficiencies and cut the cost of waste collection as well as incentive recycling. The company has developed one of the most advanced and complete devices of the sector, which includes several sensors that let the system collect other data than the fill level of trash containers [12].

There are two different models in the portfolio [4] the Enevo WE-008 for solid waste, and the Enevo WEL-001 for liquids. They use ultrasonic sensors to detect the quantity of both type of waste in a container.

In addition to the ultrasonic sensor, the device disposes of temperature and motion sensors, to automatically detect container collections and other unusual events like fire and vandalism. Regarding the connectivity, it is equipped with a penta-band GSM module to send the data and alerts to the cloud. Furthermore, Enevo has joined the LoRa Alliance and they have announced that support for the LoRa and other wireless protocols will be added to the device in future versions.

The device is packed in a body that complies with several regulations after passing corrosion, vibration and temperature resistance tests. It has a 3.6 V lithium battery with an expected lifetime of 10 years.

The value-added of Enevo is that they have built its system to be compatible with other industries, so they have already created an open API for this purpose. Thus, it is possible to use its data in other platforms.

Between November 2013 and January 2014, Enevo presented two patent applications ("Smart waste collection system and method" [5] and "Sensor device for smart waste collection systems and method" [9]), which claim the invention of a smart waste collection system, and the physical characterization of the sensor they use for this purpose.

The first one refers to the system used for smart waste management, which is based in a server system for receiving signals from one or more smart containers with one or more sensors each one. The system is able to plan the collection of the containers based on the quantity of waste found and also the quality of the material, by obtaining data from other built-in sensors like a solid state methane sensor, an electro-chemical gas concentration sensor, a pulsed hot-wire pellistor sensor, and an infra-red absorption sensor. The information is collected and processed to generate a plurality of job offers for the different waste collection companies subscribed, which would bid on some job offers to collect the waste of the specified containers.

The second patent presents the difficulties and harsh environment conditions that the sensor has to withstand. They define different environmental conditions such as high temperatures or condensation of the air, which may lead to humidity and even crystallization of the water in cold epochs. Several recommendations are claimed in this patent such as the material used for the package, or a heat reflector placed between the sensor and the lid of the container. Furthermore, in this patent they also claim the invention of a method to provide an easy and safe installation of the sensors in trash containers.

2.1.6 Urbiotica

Urbiotica is a company founded in 2008 in Barcelona that develops sensor systems to generate real time information in cities. They offer solutions for sensing parking and traffic, but also for waste management.

The device that Urbiotica offers for waste management purposes is called U-Dump M2M waste management sensor [30]. It works with ultrasonic technology to sense the occupation level of the containers, which is collected every hour. The device has a SIM on chip and a SIM slot in order to connect to the cellular network and send the collected data to the cloud. It can also send SMS if there are any connection problems through the GPRS data network.

U-Dump M2M also generates filling and emptying alarms based on the fill level it registers, and it is also able to detect the presence of fire inside the container and send an alert to the platform.

The installation of the device in a container is easy and safe, and the battery lifetime is guaranteed to be 10 years, even in the worst conditions of operation.

Figure 2.2: Urbiotica sensor installed in a waste container. Source: [30]

2.1.7 Overall Comparison

As it has been shown in this section, there are several implementations covering the main need, which is to remotely monitor the fill level of waste containers distributed by a city or town. Thus, all the products presented have two common features: a sensor for detecting the fill level of the container, and a cellular communication module.

For the implementation of the communications, all the devices use the cellular network because it is cheap and it is already deployed with a huge coverage, so it does not need additional communications systems to be designed or implemented for the basic functionality of the product. Regarding the fill level measurement, most of the products use an ultrasonic sensor for this purpose, while one of the companies uses a camera.

An ultrasonic sensor is a cheap sensor that allows measuring distances, by means of the well-known Sonar technique. It consists on emitting a pulse of sound and listening the echo generated by the target, so the distance can be obtained from the time measurement. By contrast, a camera is more expensive than an ultrasonic sensor, and the data generated is heaviest, but it is possible to extract more information from an image. This is the added value of the product offered by Compology, which, thanks to the use of a camera, it is able to identify the type of waste stored in a container, so the collection can be planned more efficiently.

Besides these common features, some companies have incorporated more sensors to their products, in order to take profit of the deployment of the system and to be able to obtain more data. These sensors are mainly intended to monitor other aspects about the current status of the container, for example a GPS receiver to precisely locate it, a temperature sensor, or even motion sensors in order to register container collections or movement. The motion sensor can also be used together with a fire detection system in order to figure out if any container is suffering vandalism, as the product of the company Enevo does.

These additional sensors that the devices embed, are the main difference between the

various products, given that all of them accomplish with the original need. In table 2.1, an overview of the main features of each product is shown, considering the built-in sensors of each system.

	Fill-level measurement	GSM	Battery Lifetime	Motion detect	Fire detect	Additional features
aEnergis (sigrenEa)	Ultrasonic sensor	Yes	5-12 years	No	No	GPS, WM-BUS
Mobile Automation	Ultrasonic sensor	Yes	10 years	No	No	N. A.
Compology	Camera	Yes	N. A.	No	No	Temperature, pH
SmartBin	Ultrasonic sensor	Yes	10 years	Tilt sensor	No	Temperature, GPS
Enevo	Ultrasonic sensor	Yes	10 years	Motion sensor	Yes	Liquid waste, methane, gas
Urbiotica	Ultrasonic sensor	Yes	10 years	No	Yes	N. A.

Table 2.1: Comparison of the main waste management sensors in the market

2.2 Ultrasonic distance measurement

In this section, it is faced the problem of how to determine the fill level of the containers by means of ultrasonic sensors. The basics on ultrasonic distance measurement are explained, and some algorithms to detect fill levels based on this technology are presented and analysed to evaluate its suitability for the purpose of the project.

2.2.1 Fundamentals

One of the most common uses of the ultrasonic transducers is the distance measurement, which technique is sometimes called also Sonar. It was invented to detect objects on and under the water by vessels, and it has been used later for air navigation and atmospheric investigations.

The transducer generates a pulse and sends it to a particular direction. When an obstacle is present in this direction, the pulse reaches it and an echo is reflected back to the transducer. By measuring the time it takes the pulse to get back to the transducer, it is possible to calculate the distance to the obstacle, as long as the propagation speed in the present medium is known. Figure 2.3 shows an overview of the operation principle.

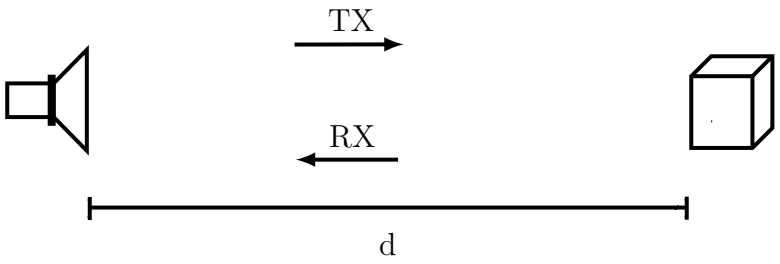


Figure 2.3: Operation principle of ultrasonic sensors

In order to determine the distance, a simple quotient is needed to translate time to distance, dividing the total elapsed time by twice the velocity of propagation in the present medium, just to take into account that the wave travels twice the distance. Therefore, the distance can be calculated as follows

$$d = \frac{\Delta t}{2\ v} \tag{2.1}$$

where d is the distance to the obstacle, ΔT is the time measured since the pulse is sent until the echo is received (usually called Time Of Flight, TOF) and v is the propagation speed of the sound over the medium. In the where case the medium is the air, the speed of the sound is $v = 343\ m/s$ (at 20 $°C$), but it increases 0.6 m/s each Celsius degree the temperature increases.

2.2.2 Project Requirements

Given the harsh environmental conditions of operation of the system, ultrasonic sensors have been selected in front of infrared or Laser sensors, which can also be used for obstacle detection and distance determination purposes. Moreover, the measurement of the container fill level presents several difficulties due to the variety of the shapes and materials of the included objects, as well as the non-uniform distribution inside the container.

Starting from the simple principle of operation of an ultrasonic range finder, there are several approaches which can help providing enough accuracy for the fill level determination.

Each one of those approaches has some advantages and disadvantages that can determine if they fulfil the requirements for the application and if they are suitable for this project.

Being conscious of what the purpose of the system is, there are some specific requirements that the fill level determination algorithm must fulfil, in order to be selected for this project.

In one hand, the main challenge is that the trash is usually distributed in a non-uniform way, which means that the system has to be able to distinguish between different fill levels within the same container. That's why instead of measuring the minimum distance to an object, it is needed to determine the distance to several zones with different distances, thus avoiding false filling measurements.

Moreover, the cost of the product, its installation and its maintenance should be taken into account, so as to design a competitive product. Therefore, a limited number of ultrasonic sensors should be used and they must be placed in a single point, in order to avoid cables under the lid of the container, which would increase the installation and maintenance costs.

2.2.3 Fill level determination algorithms

This section presents some algorithms based in ultrasonic sensors that can be used for the determination of fill levels. Moreover, Its suitability for this project is discussed by analysing which methods meet the requirements.

Radar techniques

Ultrasonic sensors are usually used to measure the distance to the closest object, causing an important loss of information of what is beyond the closest obstacle. In [2], it is presented a system that allows detection of multiple targets by means of existing radar techniques.

In that paper, an ultrasonic sensor system is designed that is capable of detecting multiple targets with a resolution of 2 cm. This is achieved by the application of radar techniques to an ultrasonic sensor. The solution proposed consists on the following techniques:

Pulse compression: Carefully design of the emitted pulse can help to narrow the autocorrelation function of the pulses, so even overlapping echoes can be separated when the received signal is filtered properly.

Optimal filtering: Use of matched filter in order to optimize the signal-to-noise ratio and take advantage of the pulse compression to resolve different targets when they are close.

Peak detection: Instead of level detection, use of peak detection of local maxima with further validation of them to avoid false detection of autocorrelation side lobes. Thus, the position estimate is amplitude independent.

In this way, an optimized pulse is sent by the transducer, and the received signal is filtered through a matched filter, thus resulting in an autocorrelation function. Then, the obtained signal is applied to a peak detector where, side lobes discarded, the local maxima are found so as to identify the distance to multiple targets. Therefore, the system provides multiple target detection, due to the resolution of multiple echoes in a single scan, allowing to detect the distance to several targets in the same direction.

Although the ability to resolve multiple obstacles would be useful to determine the fill-level of the container, it would not represent a significant advantage with a small number of sensors, since several sensors distributed among the lid of the container would be needed to accurately estimate the fill level of the whole surface.

Synthetic Aperture Focusing Technique

An ultrasound imaging system for contour detection by means of the Synthetic Aperture Focusing Technique (SAFT) is presented in [6]. Using a low number of transducers in a two dimensional array, the system is able to create a 3D image of the contour. Figure 2.4 shows the overview of the system used to obtain the contour.

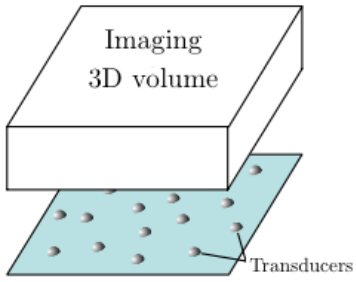


Figure 2.4: Overview of the SAFT. Source: [6]

SAFT is an ultrasonic technique which concept comes from the Synthetic Aperture Radar technique, and it has also many medical applications. The main idea is that each time, one element of the array transmits a pulse, and all the others listen to the back-scattered echo, thus forming a low resolution image of the scanned surface. By performing this operation with all the sensors of the array, and combining the data obtained from each iteration, it is

possible to obtain an image with a good resolution. Moreover, an improvement is presented in [6], which reduces the number of sensors needed to obtain a good resolution.

This technique would provide a great performance for the purpose of the project, because it is able to detect the different levels of waste distributed alongside the surface of the bin. However, it would need an important number of ultrasonic sensors, scattered under the lid of the container, which would increase significantly the cost of the product and its installation.

Multiple sensors

The measurement with a single ultrasonic sensor provides the distance to the nearest object, according to what it is stated in [14]. In that paper, it is presented the architecture, modelling, simulation and physical implementation of a system for accurate fill level estimates in common type waste bins.

As the waste usually occupies the bin space in various arrangements, the placement of a single sensor in the center of the lid typically results on an overestimation of the fill level. For this reason, combinations of two sensors were examined in [14], concluding that more accurate results were provided by placement of these sensors in a way that their areas of detection do not overlap.

By placing multiple sensors, higher resolution of the distance measurements to each independent area is obtained, and the averaging of each sensor measurement provides a more accurate result. In order to use this technique efficiently, the beam width of the sensors must be taken into account, so as to avoid overlapping of detection areas, as well as the detection of the walls, which would have a negative effect on the estimation of the fill levels.

This approach is also used in [31], where an obstacle detection system for blind pedestrian is implemented with 9 ultrasonic sensors. This system is able to provide an obstacle distribution, based on the distance measurements obtained from each sensors. Therefore, the system is able to alert the blind pedestrian informing of the obstacle location and distance.

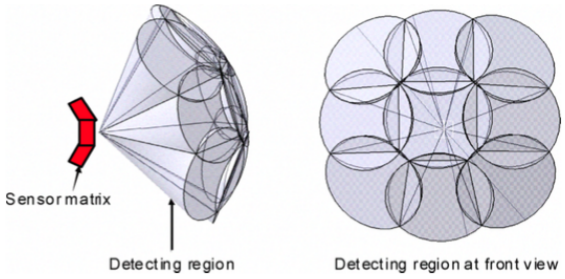


Figure 2.5: Multiple sensors approach. Source: [31]

To sum up, this technique provides more accurate estimation of the fill level, even allowing an accurate estimation of the waste distribution inside the bin. This technique suits the purpose of the project, since it allows the sensors to be placed in the same position (the center of the lid), and it helps facing the problem of the non-uniform distribution of the waste inside the container.

In this thesis, an approach with three sensors is implemented, in a try to improve the resolution by increasing the number of sensing points. For this reason, three sensors are integrated into the prototype, and a set of experiments are carried out in section 4.3 to evaluate the performance.

2.3 Hardware requirements and candidate components

In order to provide all the functions intended to be performed by the system, several components need to be integrated. This section presents these components and the requirements for each one, and also introduces some of the candidates to be included in the prototype.

2.3.1 Requirements

The main component needed in the system is a Microcontroller Unit (MCU), which has to be programmed to gather the data from all the sensors. A MCU is an integrated circuit containing a processor core, memory, and programmable input and output peripherals. MCUs are designed for embedded applications and they incorporate the necessary components to perform a small amount of instructions, in contrast with microprocessors, which are used in personal computers.

Since the system has to be installed for a long time in a fixed location, and must not need maintenance, the MCU must incorporate low power capabilities to provide long battery duration. Moreover, in order to connect to the sensors, the MCU needs also to allow communication with peripherals through different interfaces like Inter-Integrated Circuit (I2C), Serial Peripheral Interface (SPI), Universal Asynchronous Receiver-Transmitter (UART), and may also dispose of Analogue to Digital Converter (ADC) to sample the output voltage of a sensor.

A Bluetooth transceiver is also needed to provide BLE connectivity to the system. The transceiver has to comply with the BLE stack specification (without need of supporting legacy bluetooth), providing low power communication with other compliant transceivers,

by means of the standard BLE protocol stack. It is desired also that it has a low power consumption both in active and standby modes.

Another key component of the system is a GPRS module to be connected to the cellular network. The module has to support GPRS standard, and it is desirable to be quad-band capable (operating bands 850/900/1800/1900 *MHz*). So as to be integrated with the system, it needs a way to interface it with the MCU, usually UART can be used for this kind of modules.

In order to detect the fill level of the waste container, ultrasonic sensors are used by the system to measure the distance from the lid of the container to the waste. Therefore, ultrasonic sensors are also required for the system, and need to be integrated with the MCU. For this purpose, I2C buses are usually used, which let the components exchange commands and data in a master-slave way.

For the selection of the ultrasonic sensors, it is also important to take into account the beam width, the transmit power, and the sensitivity. These parameters play an important role in the capability of detection of objects, located within the beam region, with an specific section, and angle of incidence. The range of the sensor is also important, in order to cover all the height of the container.

An accelerometer is also needed in the system to detect any movement registered by the container, either due to collection processes or even vandalism acts. The accelerometer must be capable of registering the acceleration in any of the three axis, and it should also support low power mode. A specific requirement for this sensor is to be able to wake up the system from a standby mode, whenever movement in any direction is detected, usually by triggering an interrupt. This feature is commonly called *Wake on Shake* or *Wake on Motion*.

Finally, the system is intended to monitor the presence of fire inside the container, for which purpose, a flame detector is needed. Near infra-red sensors are optical equipment that responds to radiation in the spectral range of 0.7 to 1.1 μm, which is one of the most effective systems for detection of fire. In order to integrate it with the system, it is enough that it has an analogue output showing the level of radiation detected, which can be sampled by the ADC of the MCU.

2.3.2 CC2650 Wireless MCU

The CC2650 device, manufactured by Texas Instruments Inc., is a wireless MCU targeting Bluetooth Smart, Zigbee and 6LoWPAN applications. This means that it encompasses both the function of MCU and BLE transceiver.

In the technical documentation of the device [25], it is shown that it satisfies the requirements stated in this section for the MCU. On the one hand, it is a low power MCU, supporting ultra low power Sensor Controller Engine (SCE), which is a proprietary power optimized CPU that can read and monitor sensors autonomously, thus allowing the main MCU to be in standby mode. On the other hand, it supports plenty of peripherals through its GPIO ports: I2C, SPI, UART, 8-Channel ADC, Ultralow Power analogue comparator, among other interfaces. Below, Figure 2.6 shows the functional block diagram that summarizes the features of the CC2650.

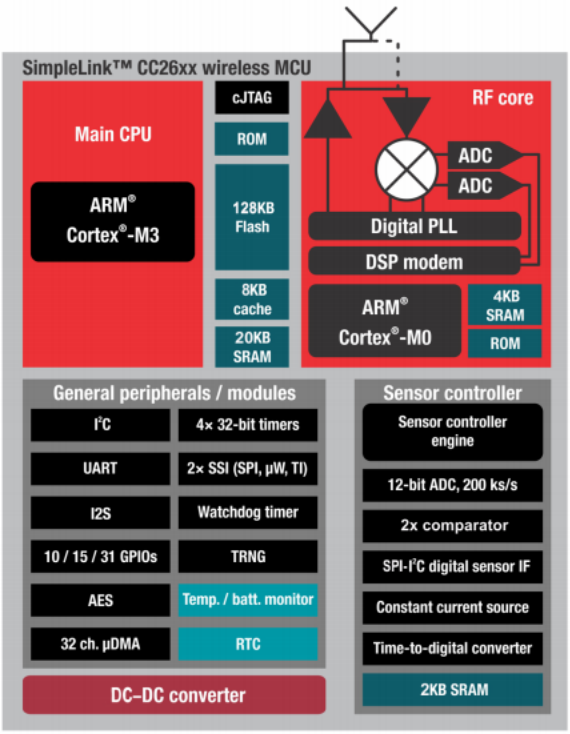


Figure 2.6: Block diagram of the CC2650. Source: [25]

Regarding the wireless interface of the CC2650, it is provided with a 2.4 GHz RF transceiver compatible with BLE 4.1 Specification and IEEE 802.15.4. The receiver sensitivity for BLE operation is -97 dBm, and it has a programmable output up to +5 dBm.

For those reasons, this device is an excellent choice; it is a 2 in 1 MCU and Bluetooth transceiver, which brings also low power functionalities that let it work with a low power consumption. In standby mode, its consumption can reach down to 1 μA; the CPU processing power consumption is 60.95 $\mu A/Hz$; and the power consumption of the radio receiving or transmitting at 0 dBm is near 6 mA, according to the technical reference manual [25].

The CC2650 wireless MCU is a Texas Instruments family device, and it runs the Texas Instruments - Real Time Operating System (TI-RTOS). This operating system is available for Texas Instruments MCU and connectivity devices as well as processors devices, providing a complete real time operating solution including protocol stacks, multitasking, multi-core communications, device drivers and power management.

In addition to the MCU and the wireless transceiver, there is a package available in Texas Instruments called SensorTag, which includes the CC2650 device together with other sensors that may be useful for the system. One of those sensors is the Invesense MPU-9250, which is a high performance accelerometer supporting the Wake on Motion (WOM) feature. These are the sensors included in the SensorTag package:

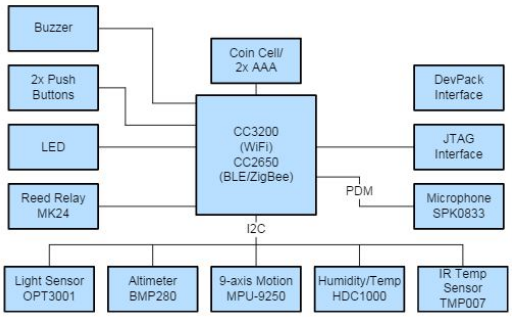


Figure 2.7: Block diagram of the SensorTag. Source: [28]

Most of these sensors were not planned to be included in the system initially, but since they are included in the SensorTag, they can be integrated eventually into the system without additional hardware required. For example, the temperature sensor could help to improve the accuracy of the fire detection system, the humidity sensor could provide useful information to determine the health of the environment, and the pressure sensor and the magnetometer could provide some information about the location of the container.

The SensorTag package also brings additional connectors like a Joint Test Action Group (JTAG) interface for programming and debugging, and a DevPack expansion connector. Through these interfaces it can be connected to the Debug DevPack, which is a small board containing a micro-USB connector and a JTAG connector. The micro-USB connector allows a computer to be connected to the SensorTag and so, programming and debugging it while the JTAG connector allows some of the CC2650 interfaces to be available in external pins, mainly to connect external sensors.

In addition to the sensors and connectors, the SensorTag also provides power supply, LED indicators and buttons. It contains a slot for placing a coin battery, thought to supply

power to the whole system at 3 V. In order to interact with the system, there are two LED indicators (red and green) that could be used as status indicators, and two push buttons located at both sides of the SensorTag.

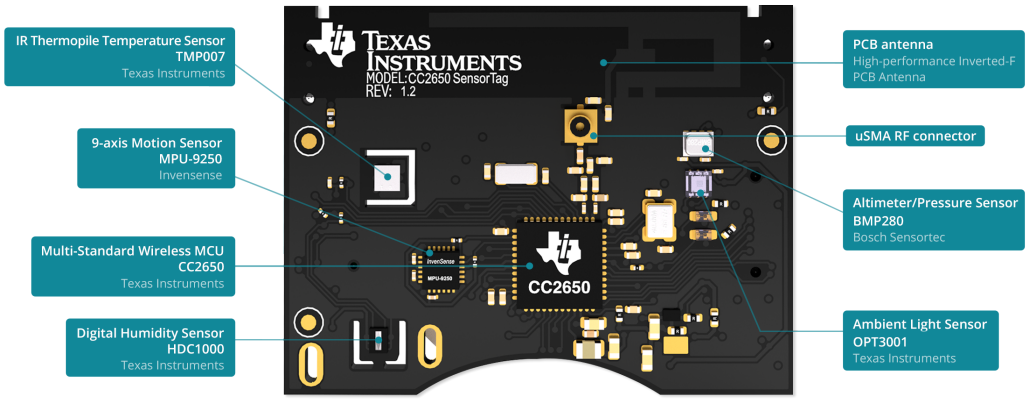


Figure 2.8: Layout of the top side of the SensorTag. Source: [28]

MPU-9250 Motion Sensor

The MPU-9250 sensor, manufactured by InvenSense Inc., is a new generation 9-axis motion sensor combining a 3-axis gyroscope, a 3-axis accelerometer and a 3-axis compass in the same chip. Moreover, this device provides low power capabilities like the Wake-on-Motion feature.

It is included in the SensorTag and it is connected to the MCU through an I2C bus working at 400 kHz, as the rest of the sensors are. However, the motion sensor has its own dedicated I2C bus, while the rest of the sensors in the SensorTag share an additional I2C bus.

Extracted from the technical documentation [8], these are some of the key features that this motion sensor includes:

- 3-Axis angular-rate sensors (gyroscopes)
- 3-Axis magnetic sensor
- 3-Axis Accelerometer with low power mode (down to 8.4 μA at 0.98 Hz)
- Wake-on-Motion interrupt for low power applications

2.3.3 Devantech Ultrasonic Sensors

There are several ultrasonic sensors in the market, which can be used for ranging purposes through measuring the Time of flight (TOF). However, not all of them use a standard protocol for integration with a MCU, which may increase the complexity of the project. That's why only sensors supporting I2C were considered for this purpose, specifically, those manufactured by Devantech Ltd. and MaxBotix Inc., two of the main manufacturers of ultrasonic sensors that present different devices supporting I2C in their portfolio.

MaxBotix Inc. manufactures the industrial I2CXL-MaxSonar-EZ sensors which are high performance ultrasonic sensors with high power output, noise rejection, auto calibration, and factory calibrated beam patterns. There are five available models with different beam widths and sensitivity each one, and the prices vary from 35 to 40 €.

Devantech Ltd. manufactures three ultrasonic sensors that support I2C bus communication, with different characteristics each one. The SRF02 is a low cost ultrasonic sensor with a narrow beam-width, the SRF08 has a wider beam width and is capable of detecting up to 17 echoes, and the SRF10 has the widest beam-width and the smallest sized transducers.

Due to the availability of SRF02 and SRF08 sensors in the department, the Devantech ultrasonic sensors have been used for the integration of the prototype, although MaxBotix devices may be tested in the future.

Devantech

As it has been already stated, Robot Electronics has three devices that support I2C in their portfolio, with different prices and characteristics each. Table 2.2 shows a detailed comparison of the features of each sensor.

	Beam-width	**Price (€)**	**# of echoes**	**# of transducers**	**Size (*mm*)**	**Others**
SRF02	Narrow	13.89	1	1	24x20x17	
SRF08	Medium	35.7	17	2	43x20x17	Light sensor Custom gain/range
SRF10	Wide	35.7	1	2	32x15x10	Custom gain Small transducers

Table 2.2: Devantech ultrasonic sensors portfolio

There is availability of one SRF08 and three SRF02 ultrasonic sensors, so both sensors are integrated in the firmware of the prototype, and their performance is compared. Three modes of operation are integrated in the prototype, which are single sensor operation, multiple sensor operation in sequential mode, and multiple sensor operation with one of the sensor transmitting and the rest just receiving the back-scattered echoes (this mode is only supported by SRF02 sensors).

The Devantech ultrasonic sensors need a 5 V source power to work, so a step-up converter is needed in the system. For the prototype integration, the 5 V source from the USB adapter in the Debugger can be used to power the sensors, but an alternative such as the Sparkfun NCP1402 5V DC-DC converter [20] has to be used in order to remove the debugger board.

2.3.4 Keyes Flame detector

The KY-026 module from *Shenzhen KEYES Robot Co. Ltd.*, is a sensor sensitive to near infrared (IR) radiation, capable of flame detection. It has an analogue output and a digital output, and a potentiometer that allows to calibrate and adjust its sensitivity. It supports to be powered between 3 and 5 V, so it can be integrated with the CC2650 MCU without any additional hardware component.

The sensor detects radiation in the range of 760 to 1100 nm wavelength, in a beam of about 60°. In order to integrate it with the system, both available outputs may be used as follows:

- Digital output can be connected to a PIN of the MCU to trigger an interrupt.
- Analogue output can be connected to a PIn to take samples with the MCU ADC.

Chapter 3

System Implementation

During the implementation of the hands-on prototype, the MCU has to be programmed to gather data from all the sensors. This chapter covers the implementation of the system, embracing the programming of the firmware for the integration of the CC2650 with the accelerometer, the ultrasonic sensors, and the flame detector.

First of all, the software architecture and the TI-RTOS (Texas Instruments Real-Time Operating System) are introduced, with a detailed description of the structure used for the implementation of the system. Then, the I2C protocol used to interface the sensors is also introduced, and finally, the driver functions programmed for the integration of each sensor are described.

3.1 Texas Instruments Real Time Operating System (TI-RTOS)

The TI-RTOS is the operating environment for Bluetooth Low Energy (BLE) projects on CC2650 devices. It is a real-time, preemptive, multi-threaded operating system that supports task synchronization. It is organized in different tasks, which are equivalent to independent threads, that are conceptually executed in parallel. In fact, only one of the tasks is running at a given time, which is selected by the BIOS scheduler based on the different priorities of each task and their status. By switching rapidly from one to the other, the feeling of concurrency is achieved.

In a BLE project, both the application and the BLE protocol stack exist as separated tasks, being the BLE protocol stack the task with the highest priority. In order to provide communication between the BLE protocol stack and the application, there is a messaging

framework called Indirect Call (ICall) used for thread-safe synchronization between them.

In one hand, the BLE protocol stack image includes the lower layers of the BLE protocol stack, from the Link Layer, to the Generic Access Profile (GAP) and Generic Attribute Profile (GATT) layers. On the other hand, the application image includes the application code, the drivers, the ICall module and the BLE profiles. Figure 3.1 provides an overview of all the entities involved:

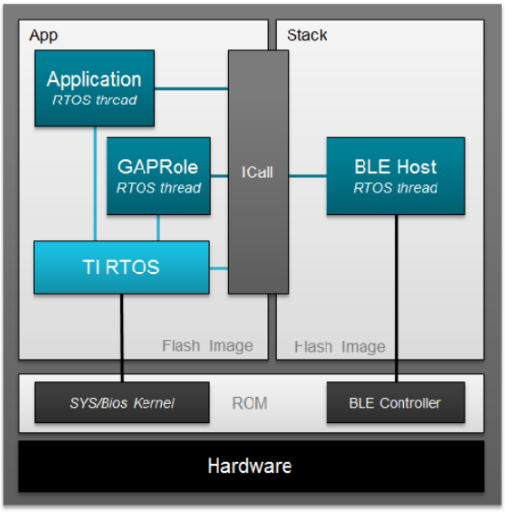


Figure 3.1: TI-RTOS top level software architecture. Source: [24]

This thesis focuses in the integration of several sensors with the MCU, which is carried out by means of the application code and drivers. This section provides an overview of some of the elements that TI-RTOS provides to develop the application project.

3.1.1 Tasks

A task is equivalent to an independent thread. One and only one is always running at a given time, even if it is the idle task. Each task is always in one of the following states:

Running: A task is being executed currently.

Ready: A task is ready to run, waiting to be scheduled.

Blocked: A task is suspended from execution, usually waiting for external resources to be available.

Terminated: A task has already terminated its execution.

Inactive: A task is in inactive tasks list.

Tasks can be blocked or terminated by several reasons, including the call of module functions, semaphores or end of execution. In any case, the processor switches to the task with highest priority of those that are ready to run. These priorities are assigned when tasks are constructed, and they are represented as numeric values, where a highest value represents a higher priority, and a lower value represents a lower priority.

The construction of tasks is performed in the main function, before the call to *BIOS_start()*, which starts the SYS/BIOS kernel scheduler to begin the execution of the tasks. When creating a task, an associated data structure defines its runtime stack for storing variables, the size of this stack, and the priority of the task. Moreover, a pointer to a task function (for example, *Task1_taskFxn()*) is passed to the *Task_construct()* function, ant this is the function that TI-RTOS will run when the task first get a chance to process.

The task function usually follows an standard structure for its proper functionality. It may have an initialization function, which is called only once, just at the beginning of the task function, and it is used to configure properly software and hardware services. Then, the task enters in an infinite loop, so that it continuously processes as an independent task and does not run to completion. Figure 3.2 shows an overview of the general structure of task functions.

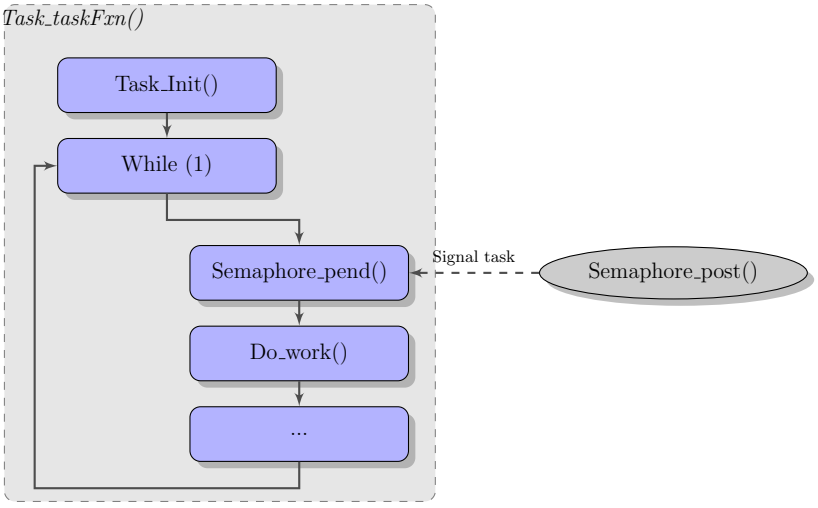


Figure 3.2: General structure of the main task function

Within the loop, the work intended to be done by the task is performed continuously. However, tasks are usually most of the time in the blocked state, where they are waiting

to an external resource or a timer to signal them. When the task is signalled, it becomes ready, and when it is executed, it goes through the loop processing the data until it becomes blocked again. Figure 3.3 illustrates the task flow just described.

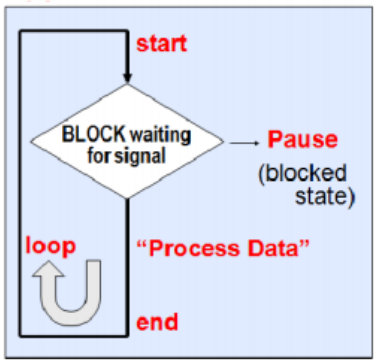


Figure 3.3: General TI-RTOS task execution flow. Source: [24]

3.1.2 Sempahores

Semaphores are one of the modules provided by the kernel package that can be used for task synchronization and mutual exclusion. They can be used either to control access to a shared resource, or to block a task until an external resource signals it.

Each semaphore stores a counter, which can be increased with the funcion *Semaphore_post()*, or decreased with *Semaphore_pend()*. This counter is a non-negative integer value that classifies semaphores in two types:

Binary Semaphore: Its counter value can be 0 or 1.

Counting Semaphore: Its counter value can be any number at 0 or greater.

Semaphore_pend() is a blocking call that checks the value of the semaphore. If its value is non-zero, it is decreased in 1 and the task continues its normal execution. However, if the semaphore counter is at 0, the task enters the blocked state and another task is started to be executed. Then, the task enters in a pending First In First Out (FIFO) queue, waiting for the semaphore to be posted. More than one task can be waiting in this queue, and they are signalled in order of arrival, without taking into account priorities, as soon as there is a call to *Semaphore_post()*.

This element is useful to block a task by pending a semaphore, until an external resource is available or needs its attention, so it posts the semaphore to signal the task.

3.1.3 Interrupts

Interrupt Service Routines (ISRs) handle critical processing that the application must perform urgently, so they have higher priorities than any task. Unlike tasks, ISRs cannot be suspended or terminated prior to completion, so they cannot do blocking calls (for example, they cannot call to *Semaphore_pend()*). There are two types of ISRs, which are Hardware interrupts (HWIs) and Software interrupts (SWIs).

HWIs respond to external asynchronous events, and they are managed through the device specific HWI modules. They are usually abstracted by means of the peripheral drivers to which they pertain to. HWIs have the highest priority of the system, even higher than the BLE protocol stack task, so it is advisable to move the processing to a handler out of the HWI context.

The second type of ISRs are SWIs, which also have higher priorities than tasks, only under HWIs. Unlike HWIs, they are preemptive and can have different priorities assigned for each SWI. For this reason, they are usually used to defer HWIs servicing to a less restrictive thread, so it is still possible to process ISRs with high priority, without blocking new HWIs to be collected, allowing them to be processed in a FIFO way as Figure 3.4 shows in an execution example.

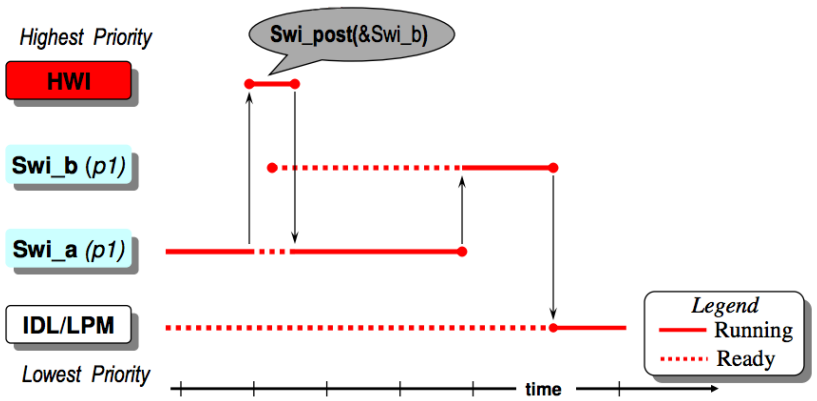


Figure 3.4: ISRs execution example. Source: [29]

In order to avoid long ISR execution times that do not allow execution of other tasks during that time, it is wise to keep a small amount of processing within an ISR context. A common use of the ISRs is to post a semaphore that is pending from another task. Thus, the ISR just signals the task, and all the processing can be performed in the task, out of the ISR context.

3.1.4 Development environment

Code Composer Studio (CCS) is an Integrated Development Environment (IDE) supporting Texas Instrument's MCUs and embedded processors portfolio. It is based on the Eclipse open source IDE, and it comprises a set of tools used for developing, compiling and debugging embedded applications.

The whole project has been developed with CCS in C language, and all the code has been uploaded to a private Git repository [17], in order to track all the changes made during the prototype implementation. CCS has been also used to flash and debug the program, through an XDS110 debug probe, provided by the Debug DevPack board.

The needed documentation for the prototype development is available at Texas Instruments website and consists mainly on the technical reference manual [26] and the developer's guide [24], which provide detailed information about all the TI-RTOS elements, including those already introduced in this section. Moreover, there is a complete wiki [29] with additional information about all the TI-RTOS components, even with online training videos. Finally, the Simplelink Academy [27] is a set of workshops that guide the developer through a practical example projects, providing a basic knowledge about different topics.

3.2 Application Project

Since this thesis focuses on the application code of the system, most of the implementation work is about to program the MCU to gather data from the external sensors, by means of the tools that the TI-RTOS provides.

As it is stated in the requirements of the project, there are three magnitudes to be monitored by the system, using different sensors. Therefore, the simplest way to divide the work is to separate the application in three independent tasks:

Movement task: Detecting any movement registered by the container.

Distance task: In charge of determining the container fill-level.

Flame detection task: Monitoring the presence of fire.

In the schematic below (Figure 3.5), the overview of the application is shown. As it can be seen, from the CC2650 kernel there are three tasks hanging, corresponding to each one of the described above. Each of the tasks communicates with one or more sensors, through an specific interface.

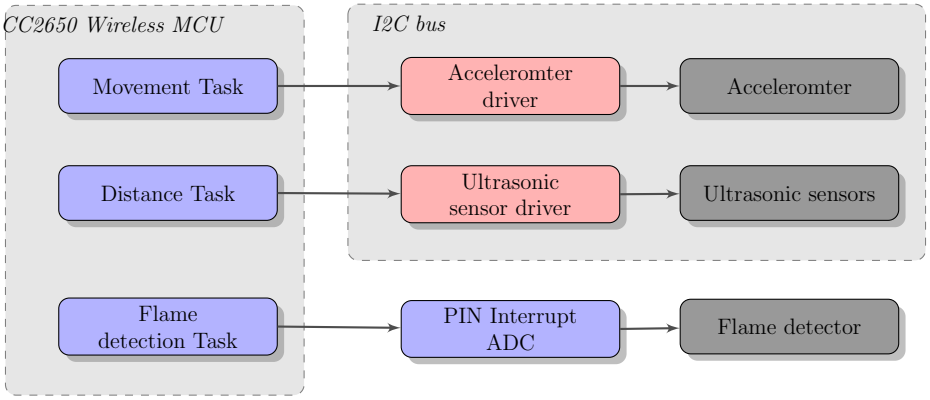


Figure 3.5: Overview of the software and hardware components of the system

All the data obtained from all sensors is intended to be reported to a remote server for later processing. Although BLE and GPRS connectivity are not available yet, all the measurements are shown in the debugger console, so as to allow evaluating and processing it. Nevertheless, this is only for the prototype evaluation, and it has to be replaced by remote reporting, once the system is endowed with connectivity.

This section focuses on the structure and functionality of each task, detailing the task configuration, the elements used by each task, and their structure. Then, section 3.3 provides a detailed description of the implementation of the I2C drivers used in movement and distance tasks.

3.2.1 Movement Task

The movement task is in charge of collecting data from the built-in InvenSense MPU-9250, which is connected to the CC2650 through an I2C bus. It is intended to be in idle mode while the device remains static, and wake up when motion is detected, taking profit of the accelerometer wake on motion (WOM) feature.

When enabled, it takes samples of the acceleration registered in each axis, so as to give live information about the movement and orientation of the container. After a given time without detecting any motion, the task should set the accelerometer in low power mode, and return again to idle mode, until the accelerometer registers any movement.

This task has a medium priority in the system, so the assigned numeric priority is 2, and the size of the stack assigned for a proper functionality is 1024 *bytes*.

The initialization function of the Movement task first initializes the necessary GPIO PINs to use the green LED and the accelerometer device. Then it sets the accelerometer

in low power mode and enables WOM feature, registering *motionInterrupt()* as the callback function for processing WOM HWIs. All the communication with the accelerometer is carried out through the functions defined in the accelerometer driver, which are described in section 3.3.2.

The main function of the movement task, *Movement_taskFxn()*, first calls the *Movement_init()* function, and then enters in an infinite loop, as usual. In this loop, the work starts by pending a semaphore, so the task is blocked until a WOM HWI signals it.

When motion is detected, the accelerometer triggers a HWI, which executes the callback function *motionInterrupt()* that posts the semaphore to signal the task. The task then switches the accelerometer interrupt mode, in order to trigger interrupts when data is ready. Thus, new interrupts can arrive either due to data ready or both data ready and WOM, so it must be checked in order to proceed.

On the one hand, if only data ready HWI is triggered, it means that the acceleration is under the threshold, so a sample of the acceleration is taken, and a counter is started in order to count the time the device does not register any motion. On the other hand, if both data ready and WOM HWIs are registered, movement over the threshold has been detected, so an acceleration data sample is taken, and the static time counter is restarted.

Moreover, every time a sample of acceleration is taken by the CC2650, the Z-axis value is checked to verify that the container is properly standing on the ground. An acceleration threshold is set as a constant in the task definition, that is compared to the measured acceleration so as to determine if it is overturned. Since the threshold is set in acceleration units, g, it is related to the vertical axis angle as follows

$$a_{th} = cos(\alpha_{th}) \tag{3.1}$$

where a_{th} is the overturn threshold set in acceleration units g, and α is the corresponding tilt angle of the container, respect to the correct orientation. In case the acceleration sample falls under the threshold, it means that the container has been overturned, and the green LED starts to blink instead of remaining fixed. Furthermore, the inactive counter is restarted, so that it never goes to sleep if the container is not properly standing on the ground.

Eventually, when the static time counter reaches a timeout, the accelerometer is set to low power mode with only HWIs enabled for WOM events, and the task is blocked by the semaphore until a WOM interrupt arrives, returning to the original state.

Figure 3.6 summarizes the flow of the movement task that has been just described.

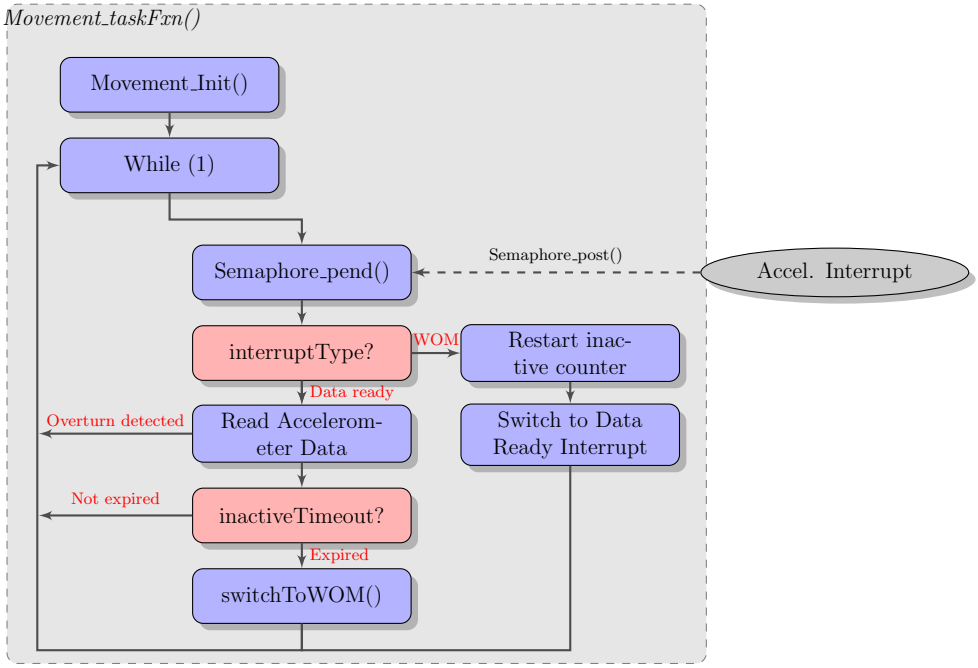


Figure 3.6: Block diagram of the movement task function

3.2.2 Distance Task

The responsible of measuring the fill level of the container is the distance task. It uses the functions programmed in the driver to communicate with the ultrasonic sensors through the I2C bus and measure the distance from the lid of the container to the trash, so the fill level of the container can be calculated.

The distance task has the lowest priority of the system, because it does not require live data and it is not a problem to delay a fill level measurement. For this reason, the numeric priority assigned to this task is the lowest, 1. The stack size assigned to this task is 2048 *bytes*.

The initialization function is only in charge of initializing the ultrasonic sensors, given that no additional PIN is used in this task. By means of the functions implemented in the ultrasonic sensor driver, which are described in section 3.3.3, the communication with the ultrasonic sensors is checked, and the number of sensors connected to the MCU is determined. With this information, and a boolean that must be set manually in the code, it is decided which of the three available ranging modes is going to be used:

Single mode: There is a single sensor and it is in charge of performing the whole ranging procedure (transmit the pulse and receive the echoes). It is selected when only one

sensor is connected.

Multiple, sequential mode: There are multiple sensors (three) and they perform independently the whole ranging procedure in a sequential way. It is selected when there are three sensors connected, and the boolean *distributed* is set to false.

Multiple, distributed mode: There are three sensors connected, and one of them is performing the whole ranging procedure, while the others are listening synchronously for the echoes. It is selected when there are multiple sensors and *distributed* is set to true (only SRF02 sensors can be set to the receive-only mode).

The implementation of these modes have been addressed for integration and performance comparison only. The final system should only include one implemented mode, which will use three sensors in the mode that it is concluded that performs better.

After initialization process is completed, and the scan mode is selected, the task function starts to perform ranging processes inside an infinite loop in the selected mode. In this task, there are not semaphores used to block the task, but only a timer is used every time a scan is performed. Figure 3.7 illustrates the task function flow.

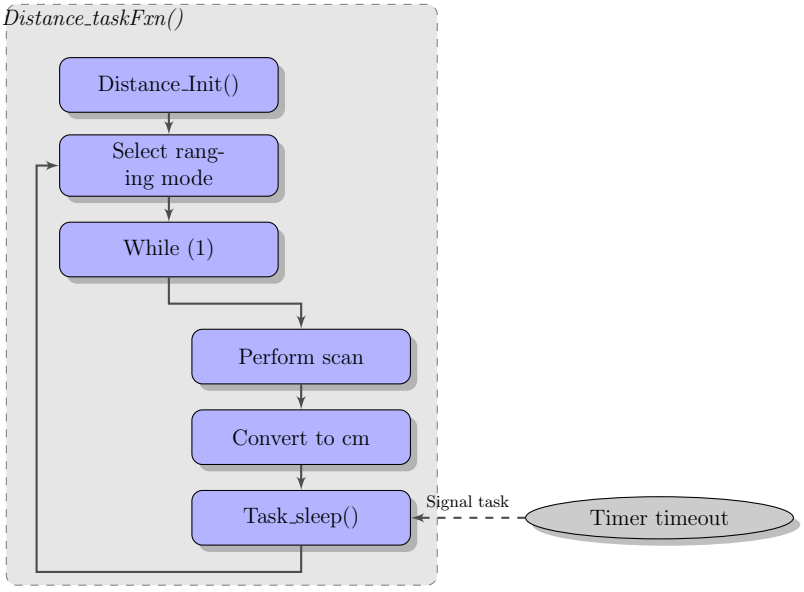


Figure 3.7: Block diagram of the distance task function

3.2.3 Flame Task

The presence of fire inside the container is monitored by the flame task. It uses both digital and analogue outputs of the Keyes flame detector in order to interact with the MCU. Similarly to what it is done with the accelerometer, the digital output of the flame detector is used to trigger an interrupt when the amount of near infra-red (IR) radiation is over a threshold.

Since this threshold can be reached because of other reasons than the presence of fire, for example direct incidence of sunlight, the analogue voltage output of the flame detector is also used. Thus, temporal samples can be taken to monitor the evolution of the near IR radiation, once an interrupt has been triggered.

This task is the most critical one because it is which needs the fastest attention, so its assigned numeric priority is the highest one, 3. The stack size of the flame detection task is set to 1024 *bytes*.

The initialization function of the flame task is in charge of configuring the PINs used for the interrupt (*Board_DP1*), the ADC (*Board_DP2*) and the green LED. Moreover, it enables the ADC and sets it up to work in PIN *Board_DP2* of the SensorTag with a manual trigger. Finally, it sets up a HWI on the positive edge of PIN *Board_DP1*, which calls the callback function *flameInterruptHandler()*.

When the task enters the infinite loop, the input value of the digital output of the flame detector is checked, and if it is low, the task is blocked by pending a semaphore. A positive edge of the digital output of the flame detector triggers an interrupt, which executes the callback function and signals the task by posting the semaphore. Then, the red LED starts to blink, and ADC samples are taken as long as the digital output of the flame detector is high. When it becomes low, the task is blocked again by the semaphore.

After some execution analysis of the flame task, it has been decided to add the collection of data from temperature and ambient light sensors, so as to provide more precise flame detection. Therefore, when the flame task is gathering data from the analogue output of the flame detector, temperature and ambient light data is obtained as well. In section 4.4, the flame detection data is evaluated and concluding remarks are extracted about the usefulness of the additional integrated data.

Figure 3.8 shows an overview of the flame detection task flow.

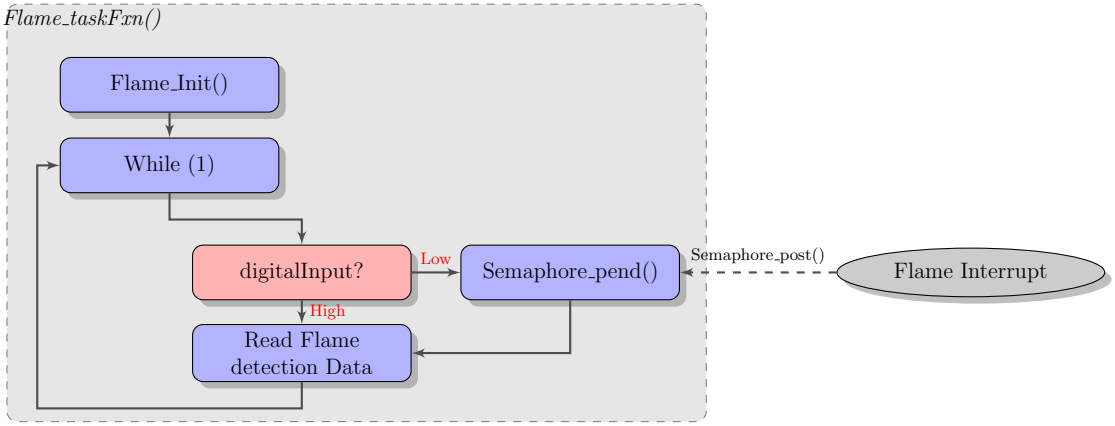


Figure 3.8: Block diagram of the flame detection task function

3.3 I2C Protocol

I2C (or I^2C, from *Inter-Integrated Circuit*) is a master-slave serial bus invented by *Philips Semiconductors* (now *NXP Semiconductors*), used to interconnect low speed integrated circuits with processors and microcontrollers, within a short distance in the same board.

3.3.1 Operating mechanism overview

The I2C bus consists on two lines, the Serial Data (SDA) is the data line, and the Serial Clock (SCL) is the clock line, used to synchronize all the data transfers over the bus. Both SDA and SCL are open-drain lines, which means that the devices can drive their output to low (G_{ND}) but they cannot drive it high (V_{DD}). Therefore, a pull-up resistor is needed in each line to allow the output to be pulled-up to V_{DD}. The overview of the I2C bus is shown in Figure 3.9.

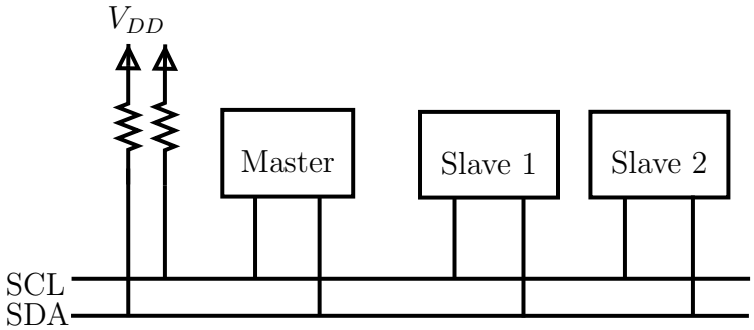


Figure 3.9: Overview of the I2C protocol connection

Although it is possible to have a multi-master bus, there is usually one only master in each bus, which drives the SCL line, and several slaves that respond to the master requests. Each slave device is identified by a 7-bit address, although some manufacturers provide it in a different format. During the operation of the transfer, the 7-bit address is sent followed by a Read "0" or Write "1" bit, thus forming a byte. That's why some manufacturers provide the slave address in a 8-bit format, where the last bit is usually the read bit "0".

The communication procedure works in a byte by byte way, and it varies depending on the operation that the master is doing, which can be either read or write. When the master is writing to any salve, the sequence is the following:

1. Master sends a start sequence
2. Master sends slave address, with R/W bit low (write bit)
3. Master sends address of the slave register that it wants to write to (a byte)
4. Master sends value to write in the slave register (a byte)
5. Master sends stop sequence

When reading from the slave, the sequence is longer and more complex:

1. Master sends a start sequence
2. Master sends slave address, with R/W bit low (write bit)
3. Master sends address of the slave register that it wants to read from (a byte)
4. Master sends a start sequence again (repeated start)
5. Master sends slave address, this time with R/W bit high (read bit)
6. Slave writes the data requested to the bus, and master reads it
7. Master sends stop sequence

The standard speed of I2C was designed to be 100 kHz in the first release, which was improved with the next update, by defining the Fast Mode supporting a speed of 400 kHz. Later, other versions have been developed supporting up to 5 MHz speeds, but such fast modes are subject to the devices specifications, and may require additional delays for the slaves to work properly.

3.3.2 Accelerometer

The MPU-9250 accelerometer comes as a buit-in sensor in the SensorTag and it is connected to the CC2650 MCU through an I2C bus, so the communication follows the protocol just described. In the case of the accelerometer, it is connected to a dedicated I2C bus, separated from the rest of the devices, which is not present in the Debug Devpack. This bus is configured to work at a clock frequency of 400 kHz, as the MPU-9250 manufacturer specifies in the datasheet [8].

In order to communicate with the accelerometer, the application project calls the functions defined in the driver file, which make the corresponding calls through the I2C protocol. The driver file for the accelerometer (sensormpu9250.c) is already provided by Texas Instruments in the SensorTag example, so it can be included in the project easily.

The address of the accelerometer is 0x68 by default, so it is used in the driver file to communicate with the accelerometer. Several functions are defined in the driver, to acquire data from the accelerometer, the gyroscope and the magnetometer, by issuing I2C commands to the corresponding registers for each purpose, whose use is described in the register map documentation [7]. In this project, only the accelerometer has been used, so these are the functions that the application uses:

sensorMpu9250Init: Initializes the accelerometer, ensuring a proper communication with the sensor, and registering the PIN used for the interrupts.

sensorMpu9250RegisterCallback: Registers the callback function for the interrupts that the accelerometer triggers when needed.

sensorMpu9250WomEnable: Enables Wake on Motion feature, and sets the threshold for motion detection.

sensorMpu9250IntStatus: Checks the type of interrupt triggered by the accelerometer. Two types are used by the application: WOM and data ready.

sensorMpu9250AccRead: Reads the values of acceleration of the three axis from the accelerometer and stores them in an array.

sensorMpu9250AccConvert: Converts raw data from the accelerometer to acceleration units g.

In addition to these functions already provided by the drivers of the SensorTag example, another function has been added to be used by the application for the purpose of this project:

sensorMpu9250SwitchInterruptMode: Switches on/off the data ready interrupt, and sets the threshold for motion detect.

This function is used in the Movement task, in order to switch from sleep mode to active mode. Thus, when motion is detected by the accelerometer, it enables the interrupt on data ready to start capturing data even if motion is not detected; when the accelerometer goes to sleep, after some time without detecting movement, it disables the interrupt on data ready so it is only triggered when motion is detected again.

3.3.3 Ultrasonic Sensors

The Devantech ultrasonic sensors have an I2C interface so as to be connected to a MCU and let it send commands and read acquired data from the sensor. The ultrasonic sensors are connected to the CC2650 through the *SDA* and *SCL* pins on the Debug DevPack board, which are connected to the I2C bus of the MCU used by all the sensors except the accelerometer. Figure 3.10 shows an image of the pin out of the Devantech SRF08 ultrasonic sensor.

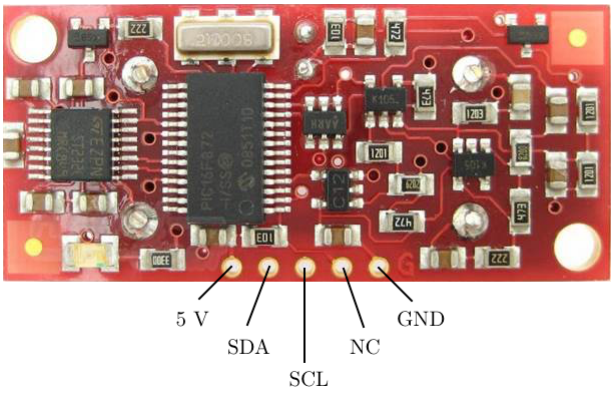


Figure 3.10: SRF08 Pin Out

There are 5 pins in the board: V_{DD} (5 V), *SDA*, *SCL*, *NC* (Not Connect), and G_{ND}. The first trouble to be solved when integrating the CC2650 with the Devantech sensors, is the different voltage levels of the devices; CC2650 is fed by a 3 V battery, while the SRF08 needs a 5 V source. The SRF08 has been tested at 3 V, but it does not respond, so an alternative solution is needed to make it work. Since the Debug Devpack is needed for connecting the sensors to the I2C bus, a 5 V source from this board is used, which comes directly from the micro USB power.

In addition to the power supply of the sensors, the voltage mismatching also affects the I2C communication. The I2C bus is a digital protocol with two possible voltage levels representing a binary signal, so if the voltage levels are different between the master and the slave, they will not be able to communicate successfully. Therefore, a level shifter is needed.

Since it is a bidirectional bus system, the level shifter must be also bidirectional. The simplest way to implement it, as it is shown in Philips documentation [13], is by connecting a discrete MOSFET to each line. In this way, the bus is divided in two sections, with pullup resistors in each section, and the devices are connected to their corresponding line. Figure 3.11 shows the configuration of the level shifter.

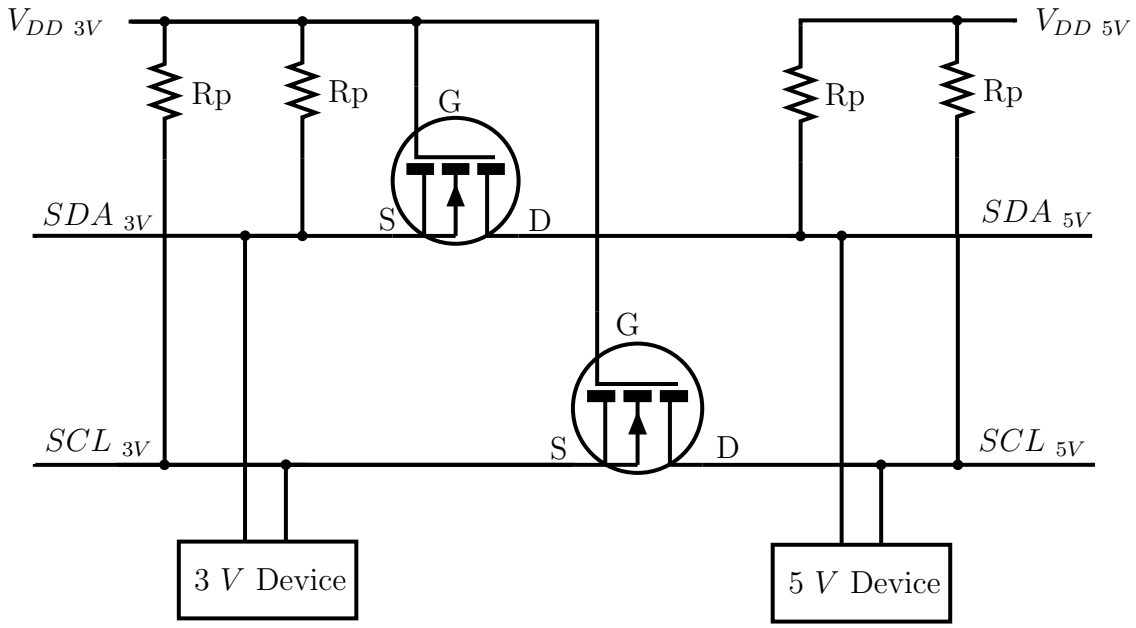


Figure 3.11: Circuit schematic of the implemented I2C level shifter

The principle of operation of the level shifter is very simple, and it can be summarized in these three possible states for each line:

State 1: No device is pulling down the line. The low voltage line is pulled up by the pullup resistor, so V_{GS} of the MOSFET is under the threshold and the MOSFET is not conducting. Therefore, high voltage section is also pulled up by its pullup resistor: Both sections are HIGH but at a different voltage.

State 2: A 3.3 *V* device pulls down the line. The low voltage line is now pulled down, which makes the source of the MOSFET be low, so V_{GS} raises over the threshold

and it starts to conduct. Thus, the line of the high voltage section is also pulled down by the MOSFET, so both sections are LOW at the same voltage level.

State 3: A 5 *V* device pulls down the line. The drain source diode starts to conduct, the low-voltage section is pulled down until V_{GS} falls bellow the threshold and the MOSFET starts to conduct. Then, the bus line of the low voltage section is further pulled down by the MOSFET, so both lines are LOW with the same voltage level.

After building the level shifter, it is needed to look at the Devantech documentation to start programming the driver functions, according to the information provided in the register map. First of all, the sensor I2C address is needed to be able to connect it to the MCU. On the SRF08 Documentation [16], it is stated that the I2C address of a factory default SRF08 is 0xE0. In fact, this is a 8-bit value (1110 0000), which means that it is provided in 8-bit format, so the last bit must be removed to be used in the CC2650 firmware:

$$\textbf{8-bit Address Sensor 1: } 1110\ 0000 = 0xE0$$
$$\textbf{7-bit Address Sensor 1: } 111\ 0000 = \underline{0x70}$$

Therefore, I2C address *0x70* is used in the CC2650 firmware to communicate with the SRF08 device. Since the algorithm chosen for the estimation of the fill level consists on three ultrasonic sensors, two more addresses are needed to connect all three sensors at the same time. The sequence of commands to change the address to a device is shown in the Devantech documentation, and it must be implemented in the driver in order to change the addresses of both additional sensors. The next two available addresses to be set to the sensors are *0xE2* and *0xE4*, in 8-bit format, according to the Devantech documentation:

$$\textbf{8-bit Address Sensor 2: } 1110\ 0010 = 0xE2$$
$$\textbf{7-bit Address Sensor 2: } 111\ 0001 = \underline{0x71}$$

$$\textbf{8-bit Address Sensor 3: } 1110\ 0100 = 0xE4$$
$$\textbf{7-bit Address Sensor 3: } 111\ 0010 = \underline{0x72}$$

Therefore, addresses *0x70*, *0x71* and *0x72* can be used for the ultrasonic sensors (no one coincides with another sensor in the SensorTag).

In order to create the driver functions, the register map must be known to read and write data from the proper registers. This information is present in the Devantech documentation, and it is summarized in table 3.1.

Location (Dec)	Read	Write
0	Software Revision	Command Register
1	Light Sensor	Max Gain Register (default 31)
2	1st Echo High Byte	Range Register (default 255)
3	1st Echo Low Byte	N/A
...	...	N/A
34	17th echo High Byte	N/A
35	17th echo Low Byte	N/A

Table 3.1: Register map of the SRF08 ultrasonic sensor

The command register accepts different values to be written with different purposes. For example, the value *0x51* corresponds to a range command in *cm* units, which is the one used in the driver functions. Moreover, a sequence of commands is defined to change the I2C address of a sensor, which consists on sending *0xA0*, *0xA5*, *0xAA* followed by the desired address to be set to the sensor.

With this information, the driver functions can be programmed. These are the functions that have been defined, and a brief description of what they do:

sensorSrf08Init: Initializes one sensor in I2C address 0x70. Returns true if it responds as expected, false otherwise.

sensorSrf08InitMultiple: Initializes three sensors, in I2C addresses 0x70, 0x71 and 0x72. Returns true only if all of them respond as expected.

sensorSrf08ChangeAddress: Issues the sequence to change the I2C address of the sensor, to the specified one.

sensorSrf08Scan: Performs a scan with a single sensor, and store the collected data in a pointer.

sensorSrf08ScanMultiple: Performs a sequential scan with three sensors, and places the data in three different pointers.

sensorSrf08ConvertCm: Converts the received binary raw data, to an array of 17 elements containing the distance to the echoes in cm.

All these functions were initially designed to work with SRF08, but they have been adapted later to work with the SRF02 as well. The only difference between using one or the other is the number of received echoes they collect. The SRF02 only provides the data for the first echo, as it does not support multi-echo scan, so the rest of the registers are empty, while the SRF08 provides up to 17 echoes.

Moreover, the SRF02 has an additional function that is not present in the SRF08, which is the fake scan command. It is issued with the command *0x56*, and it performs a scan without emitting any pulse. Therefore, it is needed that another sensor sends the pulse, so the sensor set to fake scan can listen synchronously for the echo and calculate the distance since the command was issued.

For this purpose, an additional function has been programmed to perform a distributed scan, that is available only in multiple mode (three sensors connected), and only with SRF02 performing the fake scan:

sensorSrf02ScanDistributed: Performs a scan using three sensors in distributed mode: sensor 1 performs a normal ranging procedure, while sensors 2 and 3 are notified to perform a fake scan. The results are stored in three pointers.

To sum up, the driver functions have been programmed to support all three ranging modes described before: ranging with a single sensor, and ranging with three sensors either in distributed or sequential mode. Moreover, all the functions except sensorSrf02ScanDistributed, are available to be used with both SRF08 and SRF02 ultrasonic sensors, providing versatility for the development of the tests and the final solution.

Chapter 4

Performance analysis

Once the sensors integration is finished, it is necessary to check that all the hardware components are working and performing as expected, and the data is properly gathered by the MCU.

This chapter shows the performance provided by all the system components that are integrated. First, program execution in the MCU is analysed by means of the CCS debugging tools. Then, all three external sensors are tested, so as to evaluate their performance, and determine its suitability for the system. Finally, a power analysis is carried out, to study the power consumption of each component of the system.

4.1 Program execution

As it is already explained in chapter 3, the main application project consists of three independent tasks, which are the movement, the distance, and the flame detection tasks. Each one has a different assigned priority in order to allow the BIOS scheduler to decide which task must be executed at every moment.

CCS IDE provides several tools that are useful to debug the program, and to extract interesting statistics about the performance. One of these tools is the execution graph, which represents the semaphores, interrupts and tasks executed during a short period of time.

Figure 4.1 illustrates the execution graph of first 800 μs of the initialization process of all three tasks. Although semaphores and SWIs could also be shown, it is a high memory consuming process, and with such small amount of memory that the SensorTag has, it would not allow to record an 800 μs period. Therefore, only HWIs and tasks appear in this example.

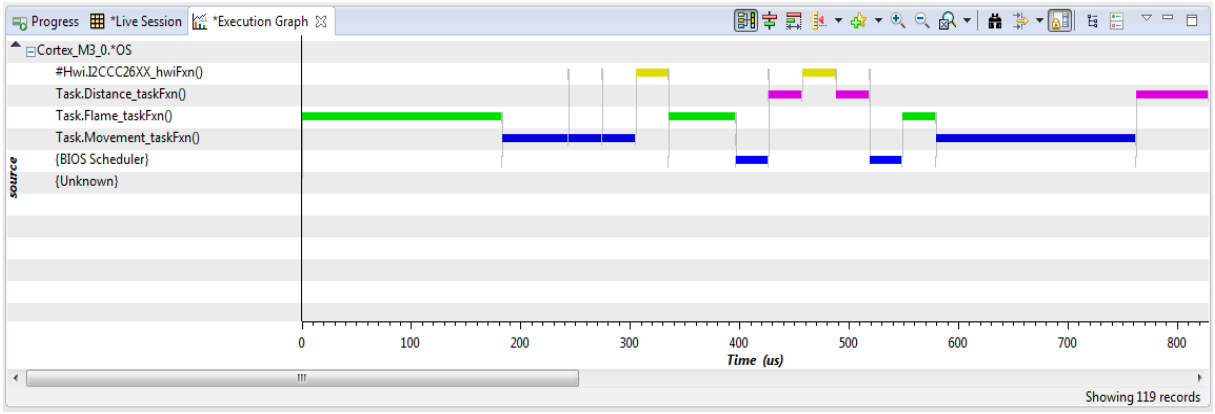


Figure 4.1: Execution graph of first 800 μs

The graphic exhibits that all three tasks are executed during the initialization process, since all of them have work to do. The preemptive behaviour of the system can be observed in the execution graph, showing a BIOS scheduler intervention before any task switching. First of all, the flame task is executed due to its high priority, until it is blocked, and then the movement task is executed. Later, the distance task is executed when both the flame and movement tasks are already blocked, waiting for external resources to signal them.

Regarding, the HWIs, it can be seen that they have the highest priority, and they are executed immediately when they are triggered, even without the intervention of the BIOS scheduler. The graph shows that all the HWIs triggered during the initialization process belong to the I2C, since WOM nor flame HWIs have been set up yet.

Another useful debugging tool is the task profiler, which generates a table with all the active tasks, and represents the percentage of time the task has been in each possible state, since it has been started. Figure 4.2 has been obtained with the task profiler during the same period of time as the previous execution graph, from 0 to 800 μs.

Progress | *Live Session | *Task Profiler

	Task	Ready Percent	Running Percent	Blocked Percent	Preempted Percent
1	Cortex_M3_0.Distance_taskFxn()	63.2	31.6	0.0	5.3
2	Cortex_M3_0.Flame_taskFxn()	0.0	21.1	0.0	0.0
3	Cortex_M3_0.Movement_taskFxn()	21.1	42.1	36.8	0.0
4	Cortex_M3_0.ti_sysbios_knl_Idle_loop__E()	100.0	0.0	0.0	0.0

Figure 4.2: Task Profiler of first 800 μs

It proves that tasks are most of the time running or ready to be running, because they are in the initialization process and they have work to do. Therefore, the idle loop is never run during this period of time. This is the main difference that can be noticed when comparing

the initialization process to the normal operation, whose task profiler during an execution time of near 10 seconds is represented in Figure 4.3.

Progress | *Live Session | *Task Profiler

	Task	Ready Percent	Running Percent	Blocked Percent	Preempted Percent
1	Cortex_M3_0.Distance_taskFxn()	97.3	0.1	2.0	0.0
2	Cortex_M3_0.Flame_taskFxn()	28.8	0.0	71.2	0.0
3	Cortex_M3_0.Movement_taskFxn()	9.3	0.0	90.6	0.0
4	Cortex_M3_0.ti_sysbios_knl_Idle_loop__E()	0.0	99.8	0.0	0.2

Figure 4.3: Task Profiler of 10 s of execution

Now, the most active task is the idle loop, while the rest of tasks are most of the time blocked, in the case of the flame and movement tasks (waiting for external HWIs to be triggered), or sleeping, in the case of the distance task (waiting to the sleep timer to expire). This is the normal case, and it demonstrates that the coexistence of multiple concurrent tasks does not suppose a difficulty for the scheduler.

Finally, the Runtime Object Viewer (ROV) is used to check the proper setting of the stack size. This view provides live state information about all the running tasks in the application. Figure 4.4 shows the ROV after 10 seconds of program execution, when already all events (including fire and motion) have happened.

address	label	priority	mode	fxn	arg0	arg1	stackPeak	stackSize	stackBase	curCoreId	affinity
0x200038f4	ti.sysbios.knl.Task.IdleTask	0	Ready	ti_sysbios_knl_Idle_loop__E	0x0	0x0	2048	2048	0x200019...	n/a	n/a
0x200032b4		2	Blocked	Movement_taskFxn	0x0	0x0	392	1024	0x200025...	n/a	n/a
0x20003214		1	Running	Distance_taskFxn	0x0	0x0	560	2048	0x200011...	n/a	n/a
0x20003264		3	Blocked	Flame_taskFxn	0x0	0x0	368	1024	0x200021...	n/a	n/a

Figure 4.4: Runtime Object Viewer after 10 s of execution

The ROV presents the priorities of each task, the size of the stack available and the peak of stack used, as well as additional information about the state of the tasks. As it can be observed, the priorities and the stack size are assigned as expected, and the peak of stack used is well below the maximum, so it can be reduced if needed

4.2 Accelerometer

The integration between the CC2650 wireless MCU and the MPU-9250 accelerometer involves two main procedures that need to be tested. On the one hand, the WOM interrupt should be triggered when the accelerometer registers any movement, which working precisely, being able to detect very tiny movements of the SensorTag. It is possible to tune the WOM

threshold between 4 *mg* and 1020 *mg*, which is useful for the system, so that detection of wind or fast vehicles passing near the container is avoided

On the other hand, after processing the WOM interrupt, the accelerometer should be switched to enable interrupts on data ready, so that the movement task can read data from the accelerometer. The data ready is also working as expected, signalling the task to get raw data from the accelerometer data register. Then, a function from the driver is used to convert raw data to acceleration units, *g*, in all three axis measurement

By default the accelerometer is set up to support a ±8 *g* acceleration range, which is enough for the application. It provides the measured acceleration in each one of the three axis, which allows to identify the orientation of the SensorTag respect to the ground. Table 4.1 shows three possible orientations of the accelerometer and their corresponding obtained output:

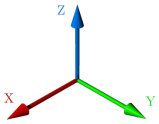
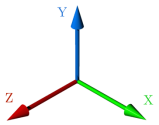
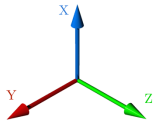
Accelerometer Orientation	**Accelerometer output**
	X: -0.0173 *g* Y: 0.0024 *g* Z: -1.0373 *g*
	X: -0.0026 *g* Y: -1.0004 *g* Z: -0.0656 *g*
	X: -0.9904 *g* Y: 0.0002 *g* Z: -0.0776 *g*

Table 4.1: Accelerometer measurements in three orientations

As it can be observed, the accelerometer shows a good accuracy, allowing to precisely determine the orientation of the container. As described in section 3.2.1, the flame task uses the value of the Z-axis to verify the correct orientation of the waste container. Currently, the application considers as a proper orientation when Z-axis acceleration is +1 *g*, which means the LEDs of the SensorTag are aiming up. By default, the overturn threshold is set to allow a 45° tilt angle, so an acceleration under 0.707 *g* is not allowed.

Although the accelerometer triggers an interrupt every time it takes a sample of the acceleration, the movement task has a customizable sampling rate that does not interact with

the accelerometer sampling rate setting. Therefore, the task has usually a slower sampling rate than the accelerometer, so the sensor is taking more acceleration samples than which are actually used. By matching both accelerometer and movement task rates, some amount of power consumption may be saved without affecting the performance of the system.

4.3 Ultrasonic Sensors

Both SRF08 and SRF02 ultrasonic sensors are subjected to some tests, so as to compare their performance. In addition to the experiments in single mode, the multiple modes (both sequential and distributed) are also explored, so their weaknesses and strengths can be compared.

The experiments are performed outdoors, in a clear environment, and with the sensors aiming upwards, so no undesirable echoes can be detected by the sensors. The selected target is a thin flat rectangular object, such as a DIN A4 folder (21x30 *cm*), held with a long stick at a known distance above the sensors.

When executing the performance comparison test between an SRF08 and an SRF02, a problem was detected affecting the SRF08 ultrasonic sensor, which is not working as expected. It performs well between 0 and 70 *cm*, being able to detect and range targets, but when the target is beyond this distance or no target is present at all, it shows a fluctuating measurement between 60 and 80 cm.

For this reason, the SRF08 test is carried out between 0 and 70 *cm*, while the SRF02 is evaluated between 0 and 150 *cm*, both in steps of 10 *cm*. In Figure 4.5, the measurements obtained with both sensors are plotted, compared to the real distance between the sensors and the target.

As it is shown, the SRF08 is capable of detecting a nearer target (varying from 3 to 5 *cm*) than the SRF02 (varying from 15 to 20 *cm*). This is due to the fact that the SRF08 has two transducers, one for transmitting and another for receiving the pulses, while the SRF02 has only one that has to be switched from transmission to reception. Thus, the minimum distance is affected by this switching time.

Regarding the overall accuracy of the sensors, the graphic shows that both measurements are quite near to the real distance, so it is presumed to be enough for the purpose of the system. Table 4.2 summarizes the statistics obtained from the performance comparison of both sensors, taking into account measurements with the SRF08 from 10 to 70 *cm*, and with the SRF02 from 20 to 150 *cm*:

Aside from the accuracy of each sensor, another important property to consider is the

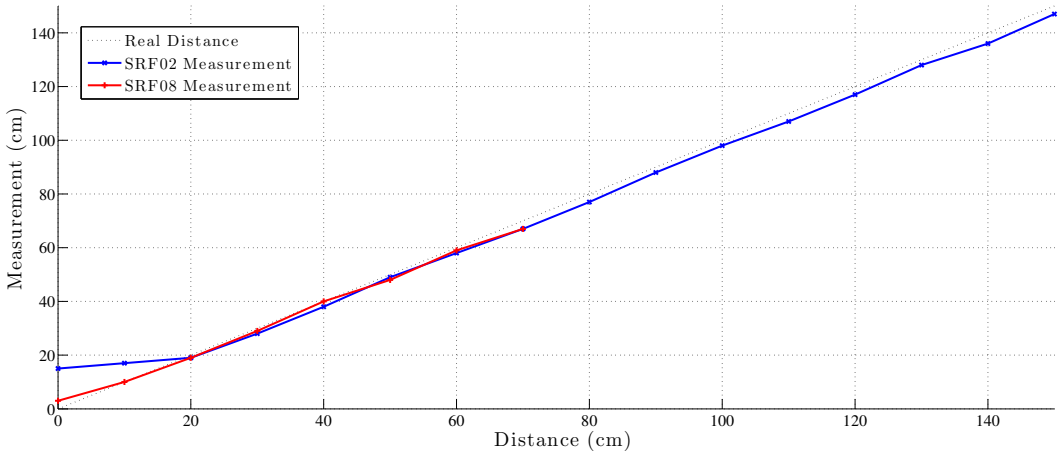


Figure 4.5: Performance comparison between SRF08 and SRF02

	SRF08	**SRF02**
Maximum Absolute Error	3 *cm*	4 *cm*
Average Absolute Error	1.14 *cm*	2.36 *cm*
Maximum Relative Error	5 %	6.7 %
Average Relative Error	2.61 %	3.28 %

Table 4.2: Summary of SRF08 and SRF02 performance comparison

impact of the different beamwidths. According to the manufacturer, the SRF02 has the narrowest beamwidth of their portfolio, while the SRF08 has a wider beamwidth, which may have two main implications.

On the one hand, a narrower beamwidth has a narrower associated detection area. Therefore, it is clear that the SRF02 sensor has more difficulties to detect a target located far from the vertical axis. As a prove of this effect, it has been measured that at a distance of 50 *cm* the Din A4 folder has to be within a 35° beam, while for the SRF08 this beam can be extended to 55° . To some extent, this could be profitable for the multiple sensors approach, so as to achieve non-overlapping detection areas.

On the other hand, when using sensors with a narrow beam to measure the range to a flat target, it exists the possibility that the echo originated by this target falls out of the beam of the ultrasonic sensor. Therefore, some thin and flat targets may be invisible to the sensor when the angle of incidence over the target surface is high. As an example, the Din A4 folder is not detected at 50 *cm* when it is centred on top of the sensor and its incident

angle is over 25°, while the SRF08 is able to detect it even at 45°.

The performance of the multiple sensors modes is also evaluated in this section. Three sensors are connected to the I2C bus, each one with an address of the ones chosen in section 3.3.3 (*0x70*, *0x71*, *0x72*). Because of the availability of three SRF02 and 1 SRF08, the tests with multiple sensors are only carried out with SRF02, both in multiple sequential and multiple distributed modes.

The same environment and target used for the previous experiments is used again, and the sensors are also evaluated from 0 to 150 *cm*, in steps of 10 *cm*. Figure 4.6 presents the results obtained from the multiple sequential mode.

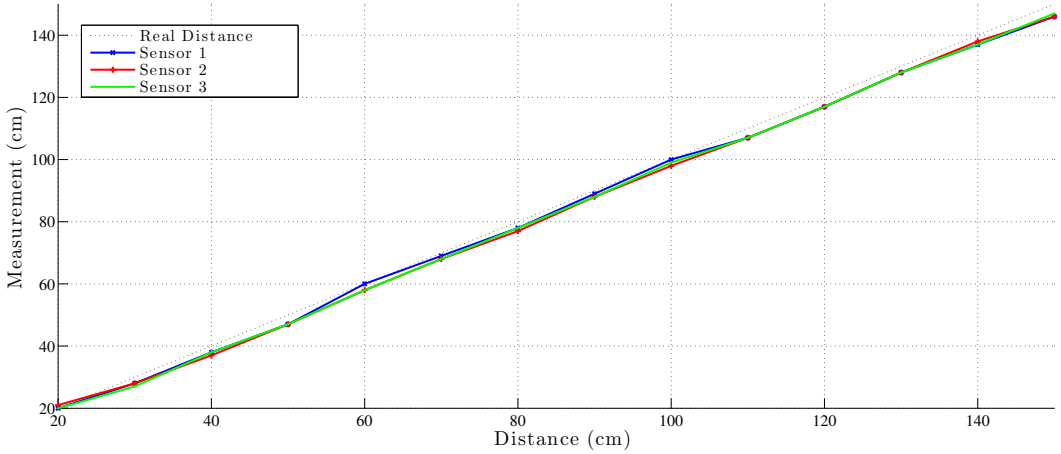


Figure 4.6: Multiple sensors in sequential mode

As expected, all three sensors have a similar accuracy, and they show similar measurements in the whole range to those obtained in the single mode for the SRF02. This is because sensors are working independently and sequentially, one after the other, so they don not interfere each other.

The multiple distributed mode is also evaluated with the SRF02 sensors. Sensor 1 sends a pulse and waits for the echo, while sensors 2 and 3 are set to fake scan mode, so they wait synchronously for the echoes of the pulse sent by sensor 1. The results of this mode are shown in Figure 4.7.

It can be observed that instead of improving the results, this mode induces an error in sensor 2 and 3. This is due to the delay on sending the I2C fake scan command to each of the sensors, and it is specially noticeable in sensor 3. It could be corrected by subtracting the delay, but this delay is not constant and there is no simple way of counting such small amount of time in the CC2650 MCU. Moreover, when the sensors would be located inside

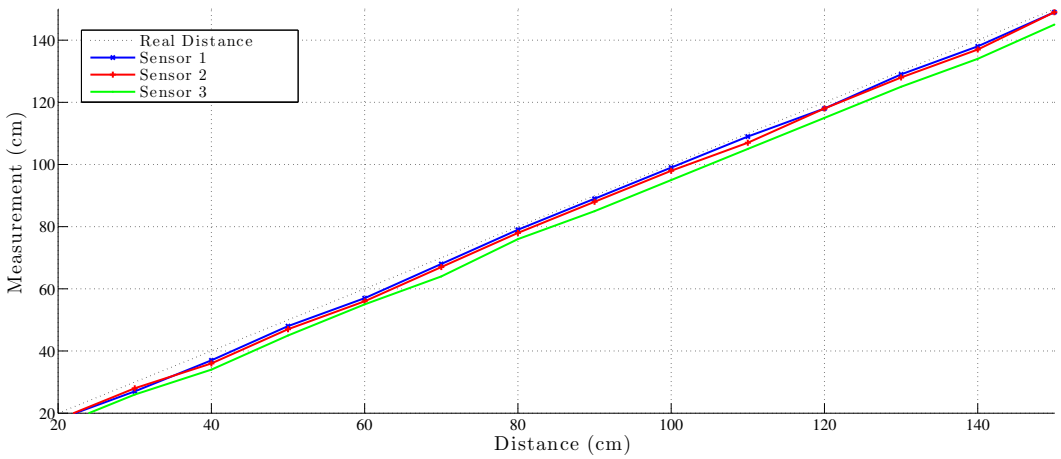


Figure 4.7: Multiple sensors in distributed mode

the container, each one pointing to a different area, the echoes would probably fall out of the beamwidth of listening sensors, and no targets would be detected by them.

The only benefit of doing the scan in distributed mode is the possibility to do more frequent scans, because there is only one echo for all three sensors, and it is not needed to wait 70 *ms* between each sensor scan. Since frequent scans are not needed for this application, the selected mode for the final system may be the sequential one instead.

As the last evaluation of the ultrasonic sensors, a concept proof has been performed with a big carton box (dimensions: 50×80×82 $cm \rightarrow$ volume: 205 l) simulating a waste container. Three different objects are used to fill the box:

Obstacle 1: A full rectangular plastic box (50×36×27 $cm \rightarrow$ 48.6 l)

Obstacle 2: An empty rectangular carton box (50×15×50 $cm \rightarrow$ 37.5l)

Obstacle 3: A folded mat (50×28×21 $cm \rightarrow$ 29.4 l)

The ultrasonic sensors are plugged in the breadboard, which is held on top of the box by two long sticks. Each sensor is aiming at a different area, with sensor 2 measuring the vertical axis, while sensors 1 and 3 are turned near 30° towards each side of the box, so as to reach the side corners. Figure 4.8 shows the layout of the experiment with all the relevant dimensions.

The total volume of the box container is 328 l, and there are three objects inside with a total volume of 115.5 l, so the fill level percentage of the box is 35.2 %. So as to estimate it

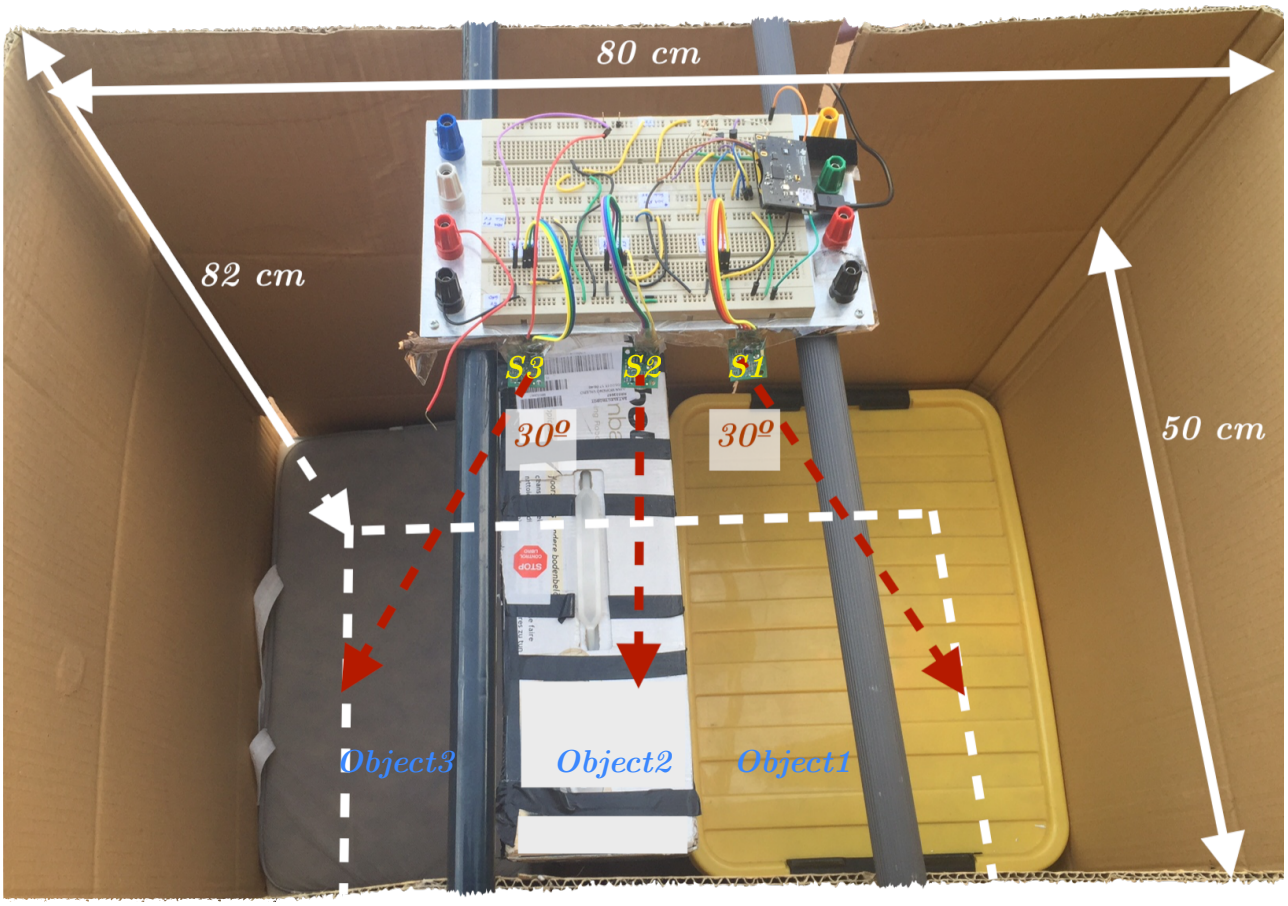


Figure 4.8: Layout of the ultrasonic sensors concept proof

with the ultrasonic sensors, the approach presented in 2.2.3 is implemented, defining three identical non-overlapping areas within the box, one for each sensor. Therefore, the volume estimated by each sensor corresponds to a rectangular prism of 50×26.7 cm surface, and the height determined by the ultrasonic sensor measurement as follows:

$$h_{meas} = h_{box} - d \cdot cos(\beta) \tag{4.1}$$

where h_{meas} is the height of the rectangular prism, h_{box} is the total height of the box container, d is the distance measured from the sensor to the object, and β is the sensor tilt angle (0° for the sensor 2 and 30° for sensors 2 and 3).

Below, it is shown the measurement from each sensor to its nearer obstacle, the height estimation given the measured distance, and the volume associated to each sensor measurement, given the scenario illustrated in Figure 4.8.

Sensor 1: $d = 58\ cm \ \rightarrow \ h_{meas} = 31.7\ cm \ \rightarrow \ V = 42.2\ l$

Sensor 2: $d = 33\ cm \ \rightarrow \ h_{meas} = 49\ cm \ \rightarrow \ V = 65.3\ l$

Sensor 3: $d = 68\ cm \ \rightarrow \ h_{meas} = 23.1\ cm \ \rightarrow \ V = 30.8\ l$

By adding the volumes of each sensor estimation, the total measured fill volume is 138.3 l, which corresponds to a percentage of 42.2 %. As expected, there is still an overestimation error, due to the low resolution of the measurements. Nevertheless, this error is much smaller than using a single sensor approach, with one only device at the center of the lid (sensor 2). In this case, the sensor detects the distance to the nearest obstacle (object 2), so the fill level estimation results in 205 l (62.5 %) which is a huge overestimation error.

Therefore, the existence of three sensing points enables an important decrease of the estimation error, but the accuracy of the system is still affected by the shape and distribution of the waste. As long as the waste is well distributed inside the container, a good fill level measurement is obtained with this approach. However, when an object with a small volume falls on top of the rest of waste, it makes the average measured height increase, resulting in an overestimation of the fill volume.

So as to prove this impact, several realizations of the previous experiment have been performed, removing one by one the obstacles from the box, and even adding a basket ball (22 cm diameter) on top of object 3. The results of the sensor measurements and fill level determination following the same algorithm, are shown in table 4.3, comparing them to the real volume.

Obstacles				**Real fill level (l) | (%)**	**Sensors Measurements (cm)**						**Measured fill level (l) | (%)**
1	**2**	**3**	**ball**		$\mathbf{d_1}$	$\mathbf{d_2}$	$\mathbf{d_3}$	$\mathbf{h_{m1}}$	$\mathbf{h_{m2}}$	$\mathbf{h_{m3}}$	
√	√	√	×	115.5 l | **35.2 %**	58	33	68	31.7	49	23.1	138.3 l | **42.2 %**
√	√	√	√	122.8 l | **37.4 %**	58	33	44	31.7	49	43.9	166.2 l | **50.7 %**
×	√	√	×	66.9 l | **20.4 %**	93	33	68	1.5	49	23.1	98 l | **29.9 %**
√	×	√	×	78 l | **23.8 %**	58	80	68	31.7	2	23.1	75.8 l | **23.1 %**
√	√	×	×	86.1 l | **26.3 %**	58	33	91	31.7	49	3.2	111.9 l | **34.1%**

Table 4.3: Fill level determination in several combinations of obstacles

On the one hand, it is observed that the worst scenario is when the basket ball is present, since it has a small volume but it increases the average measured level. In fact, the ultrasonic sensor is measuring the distance very accurately, but when the measure is averaged, it results in an overestimation respect to the real fill volume. On the other hand, the best estimation is obtained when object 2 is removed from the box, since it is the most irregular target, so the waste distribution becomes more uniform.

To sum up, the use of three sensors provides a substantial improvement compared to the single sensor approach, although the resolution is still limited. By increasing the number of narrow-beam sensors, the resolution could be increased if they are allocated properly. Besides, the scattering of sensors under the lid of the container would allow the implementation of the SAFT algorithm, which provides a better efficiency of obtained resolution in terms of number of sensor used.

4.4 Flame detector

The component in charge of detecting the presence of fire inside the container is the Keyes flame detector. It is specially sensitive to the flame spectrum and it has a detection beam of about 60°. It has an analogue and a digital output, and its sensitivity can be adjusted by means of a variable resistance.

Similarly to the accelerometer behaviour, the digital output of the sensor is used to trigger an interrupt, when the amount of near IR radiation reaches a certain threshold. When this interrupt is triggered to the MCU, the digital output starts to be checked, and the analogue output is sampled with the ADC as long as the digital one stays high. Therefore, both the interrupt functionality and the measured values have to be validated.

Regarding the digital output, it is evaluated within a dark room without any light nor window open, and the sensitivity set at its maximum value possible. In these conditions, the flame detector is able to detect the flame produced by a common lighter slightly beyond a distance of 2 m, within the beamwidth of the sensor. However, when opening slightly the window, sunlight starts to fall over the sensor and its digital output is enabled without the presence of any flame, so the sensitivity must be readjusted again to ignore the radiation coming from the sunlight. After a proper calibration, the performance obtained is the same as the one obtained before, in dark conditions.

Although inside the container it will be dark, some calibration will be needed so that a false alarms are not triggered when sunlight reaches the detector for any reason. As a first improvement of the fire detection accuracy, a temperature and an ambient light sensor that come already in the SensorTag package have been integrated into the flame detection task. Thus, when an interrupt is triggered by the flame sensor, data from the temperature and ambient light sensors is available, so it can be used to discern between fire presence or false alarms.

The inclusion of these new sensor data may allow to identify a false positive detection of fire after processing the data. Nevertheless, a good calibration is still needed so that the digital output of the flame sensor is most of the time low, thus remaining idle and saving power. So as to provide a better calibration at any time, the possibility of dynamically readjusting the sensitivity of the flame detector could be explored, providing live calibration depending on the data obtained about the environment.

Regarding the analogue output of the flame detector, it provides a voltage that varies from 0 V to V_{DD}. The highest output value is given when the sensor registers the lowest radiation, and the output decreases as the amount of near IR radiation detected grows. Moreover, the sensitivity of the sensor also affects the output value, which is proportionally decreased when the sensitivity is lowered.

The output voltage of the flame detector is monitored when its digital output is high, so it provides information about the evolution of the near IR radiation. It is also useful, together with the temperature and ambient light sensor data, to discern fire presence from false alarms. Figure 4.9 shows the output of all three sensors when a common lighter flame approaches from a distance 150 to 50 cm to the system. Blue line represents a dark environment with the sensitivity at its maximum, while red line is obtained in the same room but switching on the room light, and calibrating properly the sensitivity.

As expected, the output of the flame detector goes from 3 V to 0 V, taking different set of values for each scenario, due to the sensitivity calibration. Although any of the other sensors show significant variations during the flame approach, the ambient light sensor provides clearly different output data between the dark and the lighted scenario. Thus, this data can be used to get information about the current environment, and properly calibrate the sensor.

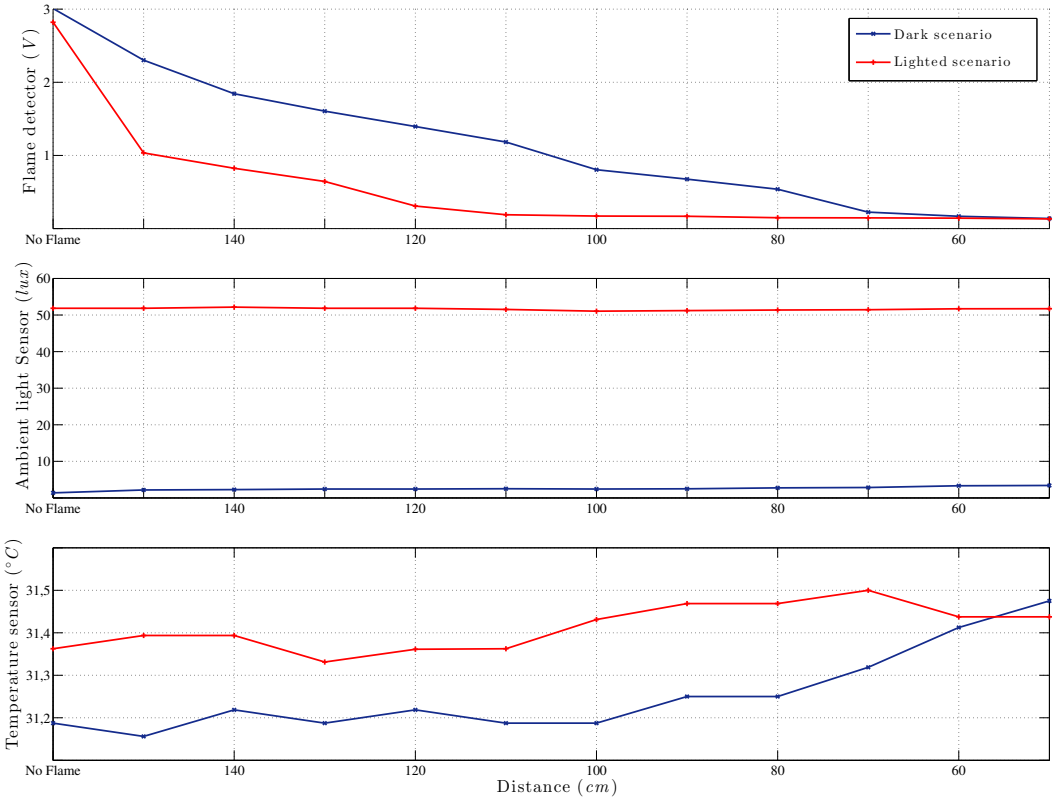


Figure 4.9: Flame task obtained data

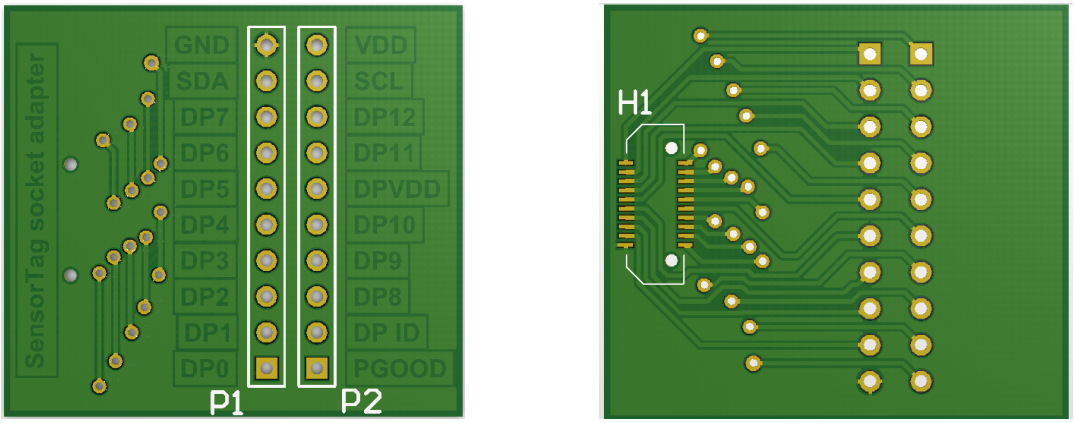
4.5 Power analysis

One important aspect of the system to be measured is the overall power consumption, so as to provide a long battery lifetime. For this purpose, a power analysis of the whole system is performed, analysing several scenarios and program executions.

Although the whole implementation was addressed with the debugger plugged to the SensorTag, it has to be removed for the power analysis execution, in order to do not take into account the Debug DevPack. During the implementation, the debugger helped to interface the MCU with the external sensors through the I2C bus and GPIO PINs, and it was also used to power the ultrasonic sensors at 5V, from the micro-USB 5V source. Therefore, an additional circuit is needed to perform these functions previously addressed through the DevPack board.

For the external sensors connection, a Samtec high speed hermaphroditic strip connector [18] is needed that allows connecting the sensors to the CC2650 I2C and GPIO PINs. A Printed Circuit Board (PCB) has been designed to adapt the connections on the Samtec

connector to a string of PINs, where all PINs available in the Debug DevPack are also accessible. Figure 4.10 shows the layout of the PCB.



(a) Top Layer (b) Bottom Layer

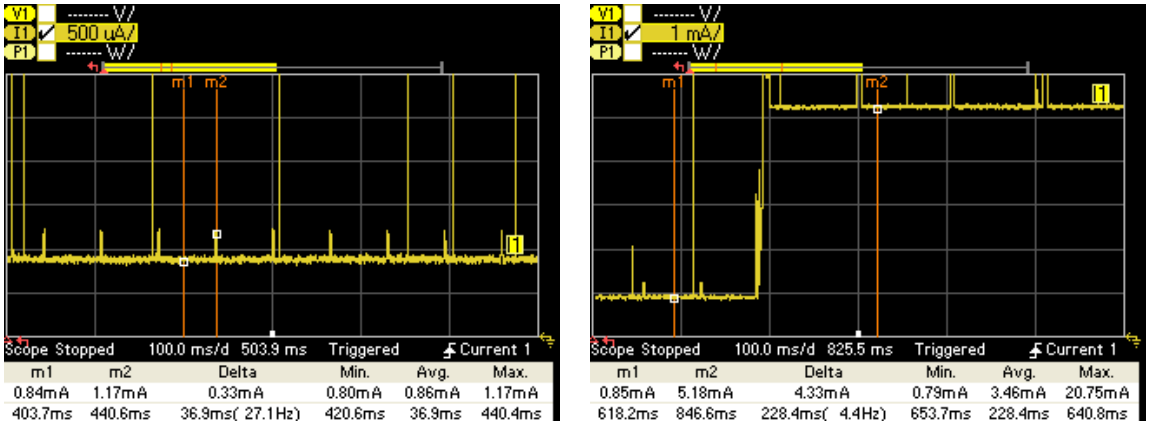
Figure 4.10: Sensortag connector adapter PCB layout

For the ultrasonic sensors sourcing, a Sparkfun 5 V step-up converter [19] is used that accepts an input within 1-4 V, and provides a 5 V output capable of driving up to 200 mA. Both the SensorTag adapter and the step-up converter can be plugged directly into the breadboard, replacing the PINs previously taken from the DevPack board (V_{DD}, SDA, SCL, G_{ND}, $Board\_DP1$ and $Board\_DP2$), by the ones in the adapter and the step-up converter. Therefore, the rest of the implemented circuit with the flame detector, the I2C level shifter and the ultrasonic sensors, remains unchanged.

A power analyser is a type of electronic instrument that allows observation of power consumption in a two dimensional plot representing the consumed power over time. To measure the consumed power, it is used as the power supply of the system, selecting the desired voltage and peak current that the system needs to work properly, and it represents in a graph the power delivered to the system at any time.

This instrument is used for the power analysis in order to measure the power consumption of the whole system in different states. First of all, only the SensorTag is connected, and the power consumption in both idle mode (a) and with the accelerometer working (b) is plotted in Figure 4.11.

It is shown that the standalone SensorTag, without any external sensors plugged, has a current consumption of 0.84 mA when it is in idle mode and the accelerometer is in low



(a) Accelerometer idle (b) Accelerometer sampling

Figure 4.11: Standalone SensorTag consumption

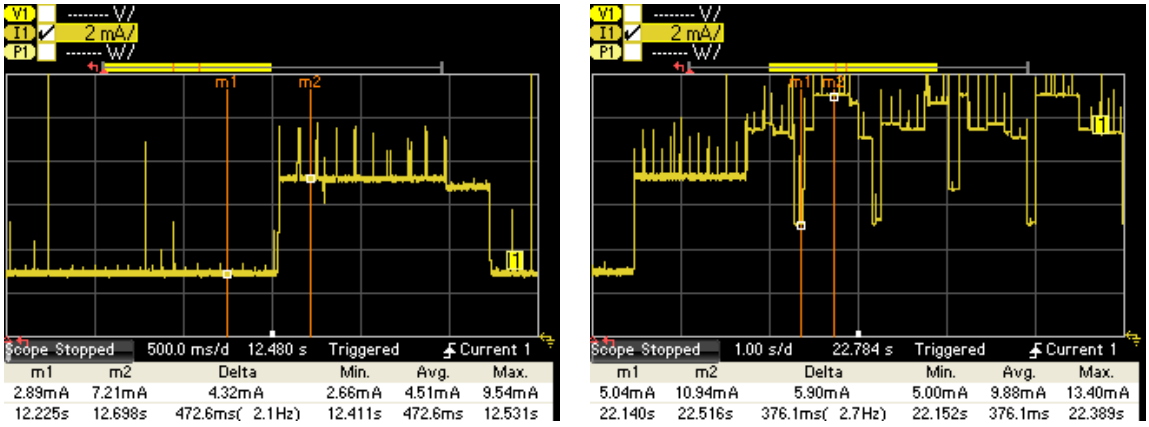
power and WOM mode. When motion is detected, the accelerometer starts to take samples and the green LED is enabled, so the current increases in 4.33 mA during this time.

Regarding the idle power consumption, it is possible to use different power profiles, which have different requirements and performance results. For example, there is an standby mode that can lower the idle consumption down to 0.34 mA, but it is not compatible with the ADC so it cannot be used with the flame detection task. An important improvement for the system could be exploring different power profiles, or even the Sensor Controller Engine (SCE) which allows performing small amounts of communications with external sensors, while the Central Processing Unit (CPU) is asleep.

While the accelerometer is enabled, there can be seen two types of power consumption events. On the one hand, there is a fixed power consumption that raises in 4.33 mA from the idle mode. It belongs mainly to the accelerometer low power mode disable, and the green LED switch on, which remain in this mode until the accelerometer goes to sleep again. On the other hand, there are several peaks of current that are periodic with a 100 ms period, and they belong to the data reading from the CC2650, which is performed with this frequency by default, but can be modified in the code. Although it cannot be appreciated in Figure 4.11, these peaks increase the current consumption in 2.1 mA.

A similar analysis is performed after connecting the Keyes flame detector, with the accelerometer also connected and running, and the results are presented in Figure 4.12.

Figure 4.12 (a) shows the power consumption with the flame detector in idle mode and the accelerometer running. As it can be seen, the consumption in idle mode raises from 0.84



(a) Accelerometer working and Keyes idle

(b) Accelerometer and Keyes working

Figure 4.12: SensorTag and Keyes flame detector consumption

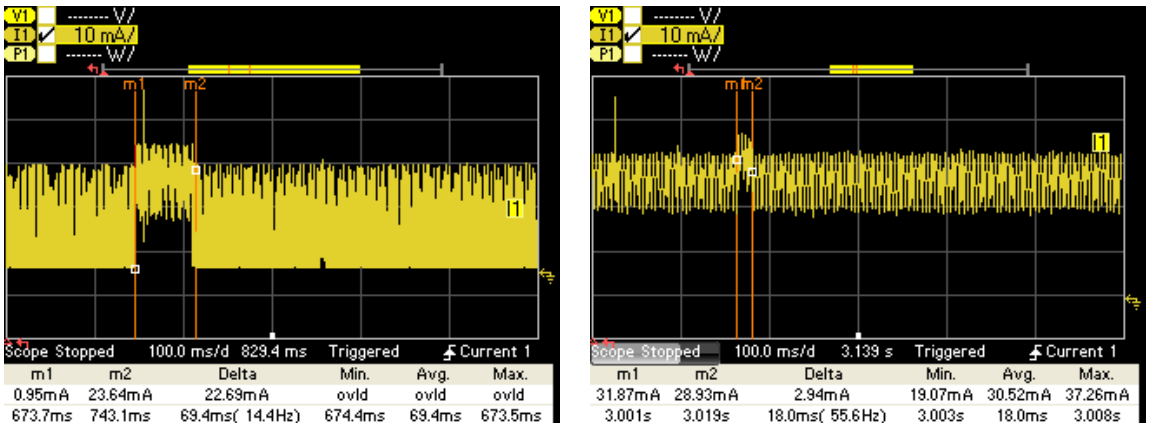
mA before to 2.89 *mA*, so the flame detector in idle mode increases the current consumption near 2 *mA*. This figure also evidences an additional behaviour of the accelerometer, which could not be appreciated in Figure 4.11 (b). Near the end of the accelerometer activity, there is an slightly decrease in the power consumption, near 0.43 *mA*), just after data peaks are finished. It corresponds to a short period of 250 ms, between the accelerometer is turned to low power mode, and the green LED is switched off. Therefore, it is clear that the accelerometer is consuming *0.43 mA*, while the green LED *3.89 mA*.

A similar experiment is illustrated in Figure 4.12 (b), where a combination of accelerometer and flame sensor is shown. First of all, the accelerometer is enabled after motion is detected, so the current consumption raises in 4.32 *mA*. Then, the presence of fire enables the flame detector, and an alternate current consumption is observed, where it increases in 2.3 *mA* first 500 *ms*, and additional 1.6 *mA* next 500 ms. This current alternation lasts until flame disappears, and similarly to what it happens with the accelerometer, is due to the red LED consumption, which for the flame tasks has a blinking behaviour, in contrast with the accelerometer.

A common peculiarity of both movement and flame tasks that have been observed in these two experiments, is that LED indicators are the responsible of most of the current consumption. In fact, the Keyes flame detector embeds two LEDs, one that is always on, and another that is switched on when the IR radiation reaches a certain threshold. Therefore, it seems that the power consumption can be lowered substantially by disabling the LED indicators, both in working and idle modes.

Finally, the power consumption of SRF08 (a) and SRF02 (b) is also studied, and it is

represented in Figure 4.13.



(a) One SRF08 sensor (b) Three SRF02 sensors

Figure 4.13: Devantech ultrasonic sensors power consumption

The first difference observed respect the accelerometer and flame sensors is that ultrasonic sensors present a current consumption with high variations. On the left, Figure 4.13 (a) shows the power consumption of an SRF08 ultrasonic sensor. It can be observed that it varies from 0 to 24 mA, when it is not within a ranging process, resulting in an average current consumption of 6.02 mA. When it is ranging, the power consumption raises 4.9 mA during almost 65 ms, which is the time corresponding to scan the maximum range of the SRF08, 11 m.

In Figure 4.13 (b), it is represented the power consumption of three SRF02 ultrasonic sensors connected at the same moment. This time, due to the presence of three ultrasonic sensors, the current consumption varies from 20.5 mA to 32.35 mA when they are in idle mode, resulting in an average of 26.41 mA (in contrast with 9.2 mA of a standalone SRF02). Within a scan process of one of the sensors, it raises 4.15 mA during almost 20 ms, which corresponds to a maximum range of less than 4 m.

The measured power consumption of the ultrasonic sensors also includes the 5 V step-up converter, which may have an important implication. The efficiency of the step up converter with all three SRF02 sensors connected is calculated below, by measuring its input and output power:

$$\left.\begin{aligned} P_{TOTAL} &= 26.41mA \cdot 3V = 79.23mW \\ P_{SENSORS} &= 12.87mA \cdot 5V = 64.35mW \end{aligned}\right\} \quad \eta_{step-up} = \frac{P_{IN}}{P_{OUT}} = \frac{64.35mW}{79.23mW} = 0.812 \rightarrow \mathbf{81.2\ \%} \tag{4.2}$$

As it can be seen, 18.8% of the ultrasonic sensors power consumption is due to the presence of the step-up converter. Therefore, it may be well reduced using ultrasonic sensors that can be fed at 3 V, like the I2CXL-MaxSonar-EZ from MaxBotix Inc. Another measure for reducing the power consumption is to enable the ultrasonic sensors only when a ranging process is going to be executed, since it will be done infrequently. Thus, the ultrasonic sensors and step up converter, which are the components that consumes the most, only would need power over a small period of time, reducing significantly the total power consumption.

Finally, an execution with all the sensors connected and working is measured, in order to see the current consumption of the whole system. Figure 4.14 shows an execution of 3s duration, where different events happen during the execution, showing the differences in power consumption.

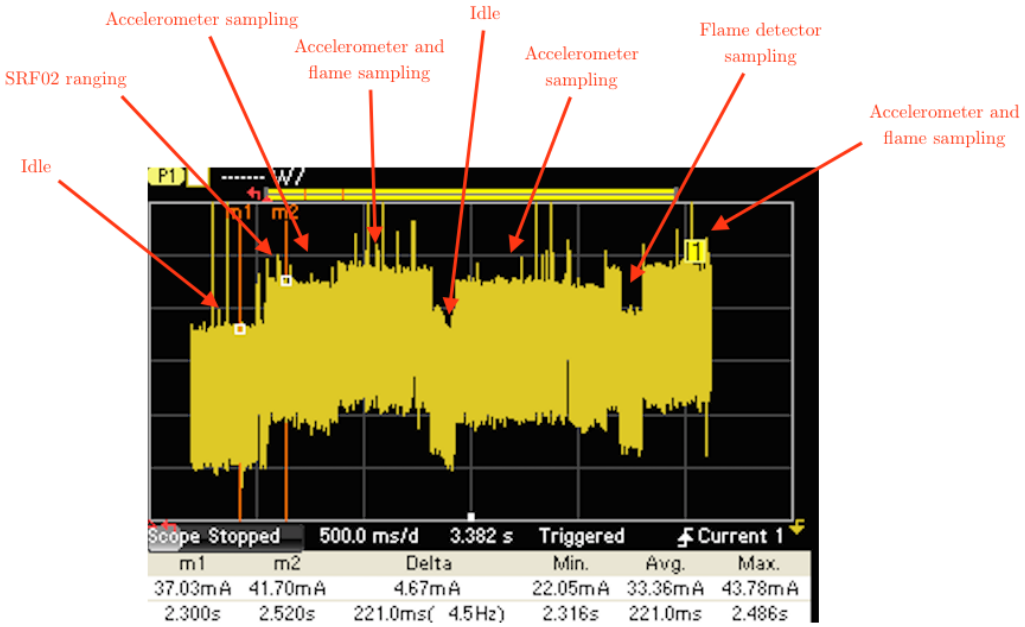


Figure 4.14: Overall power consumption in an example execution

One important observation to extract from this experiment, is the maximum absolute current consumption. It is given when all sensors are running, so presence of fire and motion are being detected, and the peak current raises up to near 45 mA. This has to be taken into account, because a coin battery CR2032, which is the default battery source of the SensorTag, usually fails when peak currents higher than 15 mA are needed. Therefore, other type of batteries such as AA batteries may be used to power the whole system.

Since the GPRS module and the BLE connectivity are not integrated still, their power

consumption is not shown in previous measurements, which will be slightly increased while data is reported. Nevertheless, given the results obtained from the power analysis of the components already integrated, some measures have been mentioned in this section that may help to significantly reduce the power consumption of the whole system. To conclude with the power analysis, those measures already mentioned are summarized:

- Different MCU configurations exists, and some of them may meet the requirements of the system while reducing the consumed power, mainly in idle mode. In addition, sensor controller engine of the CC2650 may help improving the efficiency of communications with some sensors.
- LED indicators seem to be consuming most of the power demanded by the system, when accelerometer or flame sensors are active. Since they are not needed for the purpose of the application, disabling the LEDs may be an efficient way of reducing power without affecting performance.
- In order to remove the 5 V step up converter, which is one of the inefficiencies of the system, 3 V powered ultrasonic sensors may be tested, like they are the I2CXL-MaxSonar-EZ from MaxBotix Inc.
- Currently, ultrasonic sensors are directly connected, through the step up converter, to the V_{DD} PIN that provides 3 V. However, since distance measurements will be performed from time to time, it must be changed to use a GPIO PIN controlled by the movement task, that disables ultrasonic sensors when they are not used.

Chapter 5

Budget

In this chapter, the cost of the design and implementation of the prototype developed in this thesis is calculated, taking into account the cost of the hardware components and the wages.

A breakdown of all the concepts is first shown, followed by the calculation of the total cost of the work.

5.1 Breakdown of all the concepts

The main concepts that involve an important amount of money are divided in two parts: the wages, and the main hardware components used. Small components like resistances or cables have not been taken into account in this study.

The study work has been performed by a junior engineer in part-time, during half a year. It has taken from January to June, involving 24 weeks at a part-time dedication of approximately 30 hours a week. The typical wage for a junior engineer is 15 €/h, so the total price for the worked hours can be computed as:

$$15\ €/h\ \cdot\ 30\ h/week \cdot 24\ weeks = 10800\ € \tag{5.1}$$

The other main concept of the project budget belongs to the cost of the hardware components used. Table 5.1 summarizes main components used and their associated cost:

Component	Units	Cost
SensorTag 2.0	1	26.23 €
Debug DevPack	1	13.57 €
SRF08 Ultrasonic sensor	1	35.7 €
SRF02 Ultrasonic sensor	3	13.89 €
Keyes Flame sensor	1	2.39 €
NCP1402 Step-up converter	1	7.1 €
Total		**126,66 €**

Table 5.1: Costs of the hardware components

5.2 Total cost calculation

With each concept of the cost of this work calculated independently, it is easy to obtain the total cost of the work. The result of adding the two main concepts of cost is

$$\text{Total Cost} = 10800\ € + 126,66\ € = \mathbf{10.926,66\ €} \tag{5.2}$$

Chapter 6

Environmental Impact

The system described in this thesis has a positive impact for the environment. Its main purpose is for reporting the fill levels of the waste containers, so as to plan efficiently the waste collection routes, which has two main impacts for the environment in a city or town.

First, the system will always provide an updated state of all the containers, so the collection responsible are noticed always before the containers are full. Therefore, overfilled containers are eliminated, in favour of a cleaner environment.

In the second place, it has been designed to improve the efficiency of the waste collection routes. In this way, unnecessary collections are greatly reduced, thus saving resources and CO_2 emissions.

Chapter 7

Conclusions and future work

This thesis has addressed the design and implementation of a system capable of determining the fill level of waste containers, as well as registering motion and fire detection, to fight vandalism. The measurements obtained are intended to be transmitted through the GPRS network, and the system will also be equipped with BLE connectivity, which may be used for exchanging data with nearby sensors and devices.

The work in this thesis has covered the integration of a BLE capable MCU, with external sensors to obtain the expected data. First, a review of existing similar products, algorithms to measure the fill level of a container and some candidate components for the purpose of the system have been described. Secondly, the implementation of the application project to integrate the MCU with all the external sensors have been described in detail. Lastly, both the software application and the hardware components have been subjected to an analysis of suitability for the system.

The MCU selected for the prototype has been successfully integrated with all the sensors, since it supports several interface protocols. Moreover, it provides low power configurations and BLE connectivity, which still have to be explored, but they will be very useful in future versions of the system.

Acceleration is measured by a new generation motion sensor, combining 3-axis gyroscope, accelerometer and compass. Currently, only accelerometer data is collected by the MCU, which shows a good accuracy. In addition, it has a low power consumption, and supports the WOM feature, so as to wake up the system when the it is shaken.

Two ultrasonic sensor models have been integrated into the system to be evaluated, and both of them show a good accuracy when determining the distance to a target, providing good results in a concept proof simulating a real scenario. Nevertheless, the results obtained are not conclusive, since one of them was malfunctioning, and the other one was missing

some specific targets in concrete situations. Besides, those sensors need to be powered at 5 V, which involves the need of a step-up converter, thus increasing their power consumption. An alternative ultrasonic sensor manufacturer has been mentioned that has a wider portfolio supporting 3 V power sources, which may also be studied in the future.

Flame detection is performed by a near IR radiation sensor, with a 60° beam-width. It is able of waking up the system through a digital output when a certain threshold is reached, and it has an additional analogue output that allows monitoring the amount of radiation measured. The flame sensor is capable of detecting a flame produced by a common lighter at a distance of 2 m, but its accuracy can be affected by the direct incidence of light, so it needs calibration.

So as to perform an accurate calibration of the flame sensor and avoid false fire alarms, temperature and ambient light sensors have been also integrated with the system. While the temperature does not seems to provide useful information to detect small flames, the ambient light sensor identifies the amount of light reaching the system, so it can be used to properly calibrate the flame sensor.

A power analyser has been used to measure the consumption of each component of the prototype. Several measurements are performed with different combinations of components working or idle, and the results are analysed to extract additional power saving measures. Main proposed measures are to explore the MCU power configurations, to disable all the LED indicators, to reduce ultrasonic sensor consumption by disabling them when not used, and even replacing them by 3V powered sensors.

As it has been mentioned in the scope of the thesis definition, this thesis does not involve the implementation of all the functionalities of the system. Moreover, some improvements to the already integrated services have been proposed after their analysis. Therefore, future work can be divided in two evolution lines.

On the one hand, the prototype is still missing the communications part to fulfil all the services required. For this purpose, the BLE protocol stack, the BLE profiles and the ICALL module have to be programmed and loaded to the MCU to provide BLE connectivity, and an additional module supporting GPRS has to be integrated into the system.

On the other hand, some improvements have been mentioned during the analysis of the system performance. Most of them are measures to reduce power consumption, but there are also some potential improvements related with the system accuracy and reliability. Although it is more important to finish the integration of all features in the system, these improvements should be included in future versions of the system.

Bibliography

[1] Ted Burnham. *Waste Collection Sensors: Compology*. Postscapes, First acccessed: Feb. 2016. URL: <http://postscapes.com/waste-collection-sensors-compology>.

[2] K. Audenaert; H. Peremans; Y. Kawahara; J. Van Campenhout. Accurate ranging of multiple objects using ultrasonic sensors. *IEEE International Conference on Robotics and Automation*, 1992.

[3] Jason Skylar Gates; Benjamin Chehebar. System and method for waste managment, Dec. 2014. United States Patent Application US20140379588.

[4] Enevo. *Enevo WE-008 Datasheet*, 2014. URL: <https://www.smartbin.com/solutions/ultrasonic-level-sensor-ubi/>.

[5] F. Kekalainene; J. Engstrom. Smart waste collection system and method, May 2014. WIPO Patent Application WO2014079586.

[6] O. Keitmann-Curdes; K. Hensel; P. Knoll; H. Meier; H. Ermert. 3d ultrasonic imaging and contour detection in sheet metal hydroforming. *IEEE Ultrasonics Symposium*, 2004.

[7] InvenSense Inc. *MPU-9250 Register Map and Descriptions*, Nov. 2013. RM-MPU-9250A-00. Rev. 1.4.

[8] InvenSense Inc. *MPU-9250 Product Specification*, Jan. 2014. PS-MPU-9250A-01. Rev.1.0.

[9] J. Kekalainene, F.; Engstrom. Sensor device for smart waste collection systems and method, Jul. 2014. WIPO Patent Application WO 2014114469.

[10] Moba corporation. *Smart Waste Management with Sensor Technology 4.0*, First accessed: Feb. 2016. URL: <https://mobacommunity.com/blogs/entry/Smart-Waste-Management-with-Sensor-Technology-4-0>.

[11] Kristin Musulin. *Refuse revolution: 4 companies transforming the trash bin*. Waste Dive, Sep. 2015. URL: <http://www.wastedive.com/news/refuse-revolution-4-companies-transforming-the-trash-bin/405405/>.

[12] Christoph Mller. *Smart Waste Sensors: Enevo*. Postscapes, First acccessed: Feb. 2016. URL: <http://postscapes.com/smart-waste-sensors-enevo>.

[13] NXP Semiconductors. *Level shifting techniques in I2C-bus design*, Jun. 2007. Application Note AN10441. Rev. 0.1.

[14] D. Karadimas; J. Gialelis; A. G. Voyiatzis; A. Papalambrou. A versatile scalable smart waste-bin system based on resource-limited embedded devices. *IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, 2015.

[15] Pepperl+Fuchs. *Smart Waste Management with Sensorik4.0*, First acccessed: Feb. 2016. URL: <http://www.pepperl-fuchs.com/global/en/25.htm>.

[16] Robot Electronics (Devantech Ltd.). *SRF08 Ultrasonic range finder Technical Specification*, First acccessed: Feb. 2016. URL: <http://www.robot-electronics.co.uk/htm/srf08tech.html>.

[17] Cristian Moreno Ruiz. *CC2650 Git Repository (Bitbucket)*, Jun. 2016. Owner: cristianmr.28@gmail.com.

[18] Samtec. *High Speed Hermaphroditic Strip*, Apr. 2016. LSS-110-01-F-DV-A-TR. Rev. 29 Apr 2016.

[19] Semiconductor Components Industries LLC. *200 mA, PFM Step-Up Micropower Switching Regulator*, May. 2007. NCP1402/D. Rev. 8.

[20] Semiconductor Components Industries LLC. *NCP1402 Datasheet*, May 2007. NCP1402/D. Rev. 8.

[21] Elsa Sidawy. *Sensors for trash collection in real time*. Innov'in the city, Feb. 2010. URL: <http://www.innovcity.com/2010/12/02/sensors-for-trash-collection-in-real-time/>.

[22] sigrenEa. *Connectivity for smart recycling*, First acccessed: Feb. 2016. URL: <http://sigrenea.com/es/soluciones-materiales/>.

[23] Smart Bin. *Ultrasonic Level Sensor (UBi) - Fill Level Sensor from Smartbin*, First acccessed: Feb. 2016. URL: <https://www.smartbin.com/solutions/ultrasonic-level-sensor-ubi/>.

[24] Texas Instruments Inc. *Developer's Guide*, Oct. 2010. SWRU393C. Rev. June 2016.

[25] Texas Instruments Inc. *CC2650 SimpleLink Multistandard Wireless MCU*, Feb. 2015. SWRS158A. Rev. oct. 2015.

[26] Texas Instruments Inc. *Technical Reference Manual*, Feb. 2015. SWCU117F. Rev. June 2016.

[27] Texas Instruments Inc. *Simplelink Academy v1.08*, Jun. 2016. URL: <http://software-dl.ti.com/lprf/simplelink_academy/overview.html>.

[28] Texas Instruments Inc. *Simplelink SensorTag*, Feb. 2016. URL: <http://www.ti.com/ww/en/wireless_connectivity/sensortag2015>.

[29] Texas Instruments Inc. *TI-RTOS - Texas Instruments Wiki*, June 2016. URL: <http://processors.wiki.ti.com/index.php/TI-RTOS>.

[30] Urbiotica. *U-Dump M2M waste management sensor*, First acccessed: Feb. 2016. URL: <http://www.urbiotica.com/en/product/u-dump-m2m-2/>.

[31] Yu K.H.; Yoon M. J.; Jeong G. Y. 3d detection of obstacle distribution and mapping for tactile stimulation. *IEEE International Conference on ICM*, 2009.

Glossary

M2M	Machine to Machine
GPRS	General Packet Radio Service
BLE	Bluetooth Low Energy
SIM	Subscriber Identity Module
MCU	Microcontroller Unit
WP	Work Package
WM-BUS	Wireless Meter-Bus
GPS	Global Positioning System
TOF	Time of Flight
SAFT	Synthetic Aperture Focusing Technique
I2C	Inter-Integrated Circuit
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver-Transmitter
ADC	Analogue to Digital Converter
GPIO	General Purpose Input/Output
SCE	Sensor Controller Engine
TI-RTOS	Texas Instruments - Real Time Operating System
WOM	Wake on Motion
JTAG	Joint Test Action Group
IR	Infrared
ICall	Indirect Call
GAP	Generic Access Profile
GATT	Generic Attribute Profile
FIFO	First In First Out
ISR	Interrupt Service Routine

HWI	Hardware Interrupt
SWI	Software Interrupt
CCS	Code Composer Studio
IDE	Integrated Development Environment
SDA	Serial Data
SCL	Serial Clock
MOSFET	Metal Oxide Semiconductor Field-Effect Transistor
CPU	Central Processing Unit